Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tadej Škvorc

Gručenje z omejitvami na podlagi besedil in grafov pri razporejanju akademskih člankov

magistrsko delo

magistrski program druge stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Marko Robnik Šikonja
Somentorica: prof. dr. Nada Lavrač

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Zahvala

Zahvaljujem se mentorju, izr. prof. dr. Marku Robniku Šikonji in somentorici prof. dr. Nadi Lavrač za vso pomoč in nasvete pri nastanku dela.
Zahvaljujem se Ljupču Todorovskemu, Dragiju Kocevu, Sašu Džeroskim in ostalim članom odbora ECML-PKKD 2017 za besedila člankov objavljenih na konferenci in končni urnik konference.
Zahvaljujem se svoji družini, ki mi je stala ob strani in me podpirala v času študija.

Tadej Škvorc, 2017

Contents

Povzetek

Naslov: Gručenje z omejitvami na podlagi besedil in grafov pri razporejanju akademskih člankov

Sestavljanje urnikov konference je časovno zahtevno opravilo. Urniki so sestavljeni iz različnih sej, na katerih so predstavljeni članki s skupnim raziskovalnim področjem ali podpodročjem. Ročno razporejanje člankov v urnik vzame veliko časa, saj je potrebno za vsak članek določiti, v katero področje spada. V magistrskem delu predstavimo metodo za avtomatizacijo tega postopka. Z uporabo metod strojnega učenja, obdelave naravnega jezika in analize omrežij poiščemo članke s skupno tematiko. Na podlagi podobnosti smo članke razvrstili v vnaprej definirane seje urnika z uporabo gručenja z omejitvami. Razvito metodo smo implementirali v sklopu spletne aplikacije. Za potrebe testiranja smo ustvarili podatkovno bazo znanstvenih člankov iz različnih konferenc strojnega učenja, ročno označenih z njihovim raziskovalnim podpodročjem. Vsak del metode smo testirali samostojno z različnimi pristopi, in dobili dobre rezultate. Celotno metodo smo testirali na člankih, izbranih za predstavitev na konferenci ECML-PKDD 2017. Dobili smo dobre rezultate, ki lahko služijo kot izhodišče za izgradnjo urnika konference.

Ključne besede

obdelava naravnega jezika, analiza omrežij, gručenje, organizacija konferenc

Abstract

Title: Text and Graph Based Constrained Clustering for Academic Paper Scheduling

Creating a conference schedule is a difficult task. Conference schedules consist of sessions, which contain papers that belong to the same field or subfield. Manually constructing such a schedule takes a lot of time, as each paper must be assigned to an appropriate subfield. This thesis presents a method for automating the schedule creation process. We use machine learning, natural language processing and network analysis to find papers with common research topics. Based on the similarities we group papers into predefined conference sessions using constrained clustering. We implemented the method as a part of a web application. To test the proposed method we created a database of academic papers from several machine learning conferences and labeled them manually with their research subfield. We tested each part of the method independently and obtained good results. The full method was tested on papers accepted to the ECML-PKKD 2017 conference. We obtained useful results that can be used as a starting point when creating a conference schedule.

Keywords

natural language processing, network analysis, clustering, conference organization

Poglavje 1

Uvod

Organizacija znanstvenih konferenc je časovno zahtevno opravilo. Pred začetkom konference morajo organizatorji izbrati članke, ki bodo predstavljeni na konferenci, sestaviti urnik, najti primeren prostor in opraviti še mnogo drugih opravil. Da bi organizatorjem olajšali organizacijo konferenc so se v zadnjih letih začela razvijati orodja, ki organizatorjem pomagajo organizirati in voditi konference. Ta orodja nudijo podporo pri časovno zahtevnih področjih organizacije, kot so oddajanje člankov in vodenje recenzentskega procesa, ki vodi do izbire člankov, ki bodo predstavljeni na konferenci.

Eno izmed zahtevnih opravil pri organizaciji znanstvenih konferenc je izgradnja urnika. Urniki so sestavljeni iz različnih sej, na katerih so predstavljeni članki s sorodno tematiko. Za izgradnjo urnika je potrebno poiskati članke s sorodno tematiko in jih ustrezno razvrstiti v seje urnika. Ta postopek je možno avtomatizirati tako, da najprej z uporabo metod strojnega učenja in obdelave naravnega jezika najdemo podobne članke in jih nato z uporabo gručenja ustrezno razvrstimo v seje urnika. Poleg besedila lahko za iskanje podobnih člankov uporabimo dodatne metapodatke, ki so na voljo organizatorjem, kot na primer graf citatov in podatke o izraženem zanimanju recenzentov pri izbiri člankov, ki bi jih želeli recenzirati.

V magistrskem delu predstavimo metodo, ki avtomatsko razvrsti članke v urnik. Podobne članke najdemo z uporabo različnih metod obdelave na-

ravnega jezika in analize grafov. Podobne članke razvrstimo v ustrezne seje z algoritmom za gručenje z omejitvami, ki upošteva velikosti sej in strukturo urnika. Pomagamo si tudi z metodami analize omrežij, s katerimi pridobimo dodatne informacije iz podatkov v obliki grafov.

Dodaten problem, ki ga je potrebno rešiti, je evalvacija uporabljene metode. Za kakovost urnikov konference ne obstajajo splošno uveljavljene metode evalvacije in kriteriji kakovosti, zaradi česar je težko objektivno oceniti kakovost urnika. Zato kakovost urnika ocenimo s kombinacijo različnih metod. Najprej ocenimo delovanje posameznih metod, ki jih uporabimo za samodejno razvrščanje člankov v urnik. Ker za gradnjo urnika uporabimo metode strojnega učenja, lahko za evalvacijo teh metod uporabimo uveljavljene evalvacijske metode, kot so klasifikacijska točnost, specifičnost in senzitivnost. Za potrebe takšne evalvacije smo zgradili podatkovno bazo ročno označenih člankov. Oceno kakovosti končnega urnika smo dobili s pomočjo mnenj strokovnjakov. Razvit algoritem smo testirali na člankih, izbranih za predstavitev na konferenci ECML PKDD septembra 2017 v Skopju.

V 2. poglavju predstavimo obstoječe tehnologije in postopke, ki se nanašajo na magistrsko delo. V 3. poglavju predstavimo naš pristop k problemu in predstavimo spletno aplikacijo, ki organizatorjem konferenc omogoča samodejno razporejanje člankov v vnaprej definirano strukturo urnika. V 4. poglavju opišemo kako smo izdelano metodo testirali in predstavimo rezultate. V 5. poglavju predstavimo sklepne ugotovitve in možne izboljšave.

Poglavje 2

Pregled področja

Za samodejno izgradnjo urnika konference smo uporabili kombinacijo pristopov z različnih področij strojnega učenja. V poglavju najprej predstavimo pristope za obdelavo naravnega jezika, s katerimi lahko poiščemo članke s podobno vsebino. Nato predstavimo metode za gručenje in nadzorovano učenje, s katerimi podobne članke razvrstimo v gruče. Predstavimo metode za analizo grafov, s katerimi iz grafov razberemo koristne informacije. Nazadnje predstavimo obstoječa orodja za upravljanje konferenc.

2.1 Gručenje in nadzorovano učenje

Nadzorovano učenje je eno izmed glavnih področij strojnega učenja. Cilj nadzorovanega učenja je iz vhodnih podatkov z znanimi razredi zgraditi (se naučiti) napovedni model, ki pravilno napove razrede za podatke, katerih razred ni vnaprej znan. Gručejne se od nadzorovanega učenja razlikuje v tem, da nimamo podatkov z znanimi razredi in ne zgradimo napovednega modela. Namesto tega želimo iz množice primerov najti take primere, ki so si med seboj podobni in jih združiti v gruče. Za gručenje in nadzorovano učenje obstajajo različne metode.

2.1.1 Nadzorovano učenje

Za nadzorovano učenje obstaja veliko različnih pristopov. Pristopi predpostavijo, da je vsak podatek opisan z množico spremenljivk $x_1, x_2, ..., x_n$, ki jim pravimo atributi, in s spremenljivko y, ki jo želimo z modelom napovedati. Modeli za nadzorovano učenje se delijo na klasifikacijske in regresijske glede na tip spremenljivke y. Pri klasifikacijskih je y diskretna vrednost z omejeno domeno, pri regresijskih pa je y številska vrednost in domena ni nujno omejena.

Najpreprostejši model nadzorovanega učenja je linearna regresija [50], ki izrazi vsak y_i kot $y_i = \theta_1 x_{i1} + \theta_2 x_{i2} + ... + \theta_n x_{in} + \epsilon_i$, kjer so θ_i parametri modela, x_i atributi, ϵ_i pa napaka. Regresijski model zgradimo tako, da poiščemo vrednosti $\theta_1, \theta_2, ..., \theta_n$, ki minimizirajo $\sum_i |\epsilon_i|$ ali $\sum_i \epsilon_i^2$. Slabost linearne regresije je v tem, da predpostavi, da lahko y_i izrazimo z linearno vsoto $x_{i1}, x_{i2}, ..., x_{in}$. Podoben pristop je polinomska regresija, ki predpostavi polinomsko odvisnost. Drugi pogosto uporabljeni pristopi za nadzorovano učenje so logistična regresija [25], odločitvena drevesa [46], klasifikacijska pravila [22], metoda podpornih vektorjev [24], k-najbližjih sosedov [41], naivni Bayesov klasifikator [45] in naključni gozdovi [10].

2.1.2 Gručenje

Ker pri gručenju nimamo podatkov z vnaprej znanimi razredi, se metode za gručenje razlikujejo od metod nadzorovanega učenja. Namesto, da bi vhodnim podatke želeli napovedati pripadajoči razred, želimo podobne podatke združiti v skupno gručo.

Za gručenje obstaja več različnih pristopov. Eden izmed najstarejših je hiearhično gručenje [26], ki na začetku vsak podatek razvrsti v lastno gručo in nato združuje gruče, dokler ne doseže želenega števila gruč. Metode hiearhičnega gručenja so počasne, zaradi česar so slaba izbira pri velikem številu podatkov. Hitrejša je metoda k-voditeljev (k-means clustering) [23]. Metoda k-voditeljev najprej naključno razporedi voditelje $\{c_1^0, c_2^0, ..., c_n^0\}$ v

prostor, ki ga definirajo podatki. Nato vsak podatek priredi voditelju, ki mu je najbližje, in posodobi vsakega voditelja tako, da se nahaja na sredini podatkov, ki mu pripadajo. Koraka ponavlja, dokler se voditelji ne nehajo premikati. Metoda deluje hitro tudi na velikih podatkovnih množicah. Med druge pogosto uporabljene metode sodijo še DBSCAN [14], mean shift [19] in affinity propagation [18].

Poleg splošnih metod gručenja obstajajo tudi bolj specifični pristopi. Primer je gručenje z omejitvami [42], kjer primerom pred gručenjem postavimo posebne omejitve. Tipično za nekatere pare primerov definiramo, da se ne smejo pojaviti v isti gruči, ali pa da se morajo pojaviti v isti gruči. To je koristno takrat, ko že pred gručenjem vemo nekaj o učnih primerih in bi želeli, da gručenje to upošteva.

2.2 Obdelava naravnega jezika

Obdelava naravnega jezika je področje umetne inteligence, katerega cilj je obdelava in razumevanje človeškega jezika z računalniki. Področje je široko, v grobem se deli na dva dela:

Sintaktična analiza se ukvarja s procesiranjem besedil s sintaktičnega vidika. Pogosti problemi, ki jih rešuje, so tokenizacija (razgradnja besedila v posamezne besede ali besedne zveze), oblikoslovno označevanje (part-of-speech tagging), kjer je cilj ugotoviti vlogo (na primer glagol, samostalnik, pridevnik ...) posamezne besede v stavku, luščenje terminologije in lematizacija besed. Cilj sintaktične analize je pretvoriti vhodno besedilo v obliko, ki je primernejša za računalniško analizo.

Semantična analiza se ukvarja z razumevanjem besedila. Večinoma moramo pred semantično analizo opraviti sintaktično analizo, s katero besedilo pretvorimo v obliki, ki olajša semantično analize besedila. Nekateri problemi, ki jih rešujemo s semantično analizo so strojno prevajanje, strojno odgovarjanje na vprašanja uporabnikov, analiza sentimenta, klasifikacija besedil in označevanje udeleženskih vlog.

Obdelava naravnega jezika pokriva tudi področja, kot so prepoznava govora, izgradnja povzetkov in luščenje informacij iz besedil.

2.2.1 Sintaktična analiza

V sintaktično analizo spadajo metode, ki pretvorijo vhodno besedilo v obliko, ki je bolj primerna za nadaljnjo računalniško obdelavo. Ena izmed osnovnih metod sintaktične analize je tokenizacija ali segmentacija besed. Cilj segmentacije besed je razdeliti vhodno besedilo na posamezne besede. Pri Najpreprostejšem pristopu besede ločimo glede na presledke, vendar to ni vedno dovolj. Nekatere besede, kot so na primer *nenamerno*, je včasih bolje razdeliti v dve besedi (*ne* in *namerno*), besede ločene z vezaji pa obdržati skupaj (na primer mesto Port-au-Prince), saj s tem iz besedila lažje razberemo pomen. Nekateri jeziki, kot na primer kitajščina, ne uporabljajo presledkov. Zaradi tega različna besedila potrebujejo različne pristope za tokenizacijo. Poseben primer so objave na družbenih omrežjih, kjer objave pogosto vsebujejo slovnične napake in emotikone, kar oteži tokenizacijo in zahteva pristope specifične za take objave [37].

Lematizacija je pogosto uporabljena metoda sintaktične analize. Cilj lematizacije je pretvorba besed z isto lemo (osnovno obliko) v skupno besedo (na primer pretvorba besed *hodim*, *hodimo*, *sem hodil* v *hoditi*). Podobna metoda je odstranjevanje končnic (stemming), kjer odstranimo pogoste končnice. Postopek deluje dobro v angleškem jeziku, kjer se različne oblike besede največkrat razlikujejo le v končnici (na primer besede *walking*, *walked*, *walks* in *walker*).

Oblikoslovno označevanje (part-of-speech tagging) je pomemben del sintaktične analize. Oblikoslovni označevalniki kot vhod dobijo zaporedje besed, ki jim napovejo besedne vrste (glagol, samostalnik, pridevek itd.). Poleg besedne vrste oblikoslovni označevalniki pri slovenščini določijo tudi besedne lastnosti kot sta spol in sklon. Oblikoslovno označevanje je zahtevno, ker lahko ista beseda nastopi v različnih besednih vrstah, zaradi česar slovar oznak za vsako besedo ni dovolj. Za oblikoslovno označevanje obstaja več pristopov.

Izmed pogosto uporabljenih pristopov v angleščini je uporaba skritih markovskih modelov [9]. Pri oblikoslovnem označevanju s skritimi markovskimi modeli definiramo vhodno besedilo kot zaporedje n besed $W = w_1, w_2, ..., w_n$. Napovedati želimo najverjetnejše zaporedje besednih vrst $T = t_1, t_2, ..., t_n$, ki pripada vhodnemu zaporedju besed. To lahko zapišemo kot:

$$\hat{T} = \arg\max_T P(T|W).$$

To po Bayesovem pravilu lahko zapišemo kot:

$$\hat{T} = \arg\max_T \frac{P(W|T)P(T)}{P(W)}.$$

Ker je $P(W)$ enak za vsak T lahko to zapišemo kot:

$$\hat{T} = \arg\max_T P(W|T)P(T).$$

Skriti markovski modeli predpostavljajo:

- Verjetnost, da se pojavi posamezna beseda, je odvisna le od njene besedne vrste in ni odvisna od ostalih besed in besednih vrst.
- Verjetnost, da se pojavi posamezna besedna vrsta, je odvisna le od i prejšnjih besednih vrst. Pogosto velja $i = 1$. V tem primeru gre za bigramski skriti markovski model.

S tema predpostavkama lahko enačbo zapišemo kot:

$$\hat{T} = \arg\max_T \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1}).$$

Skriti markovski model zgradimo tako, da na veliki količini ročno označenega teksta izračunamo verjetnosti $P(w_i|t_i)$ in $P(t_i|t_{i-1})$. Optimalno zaporedje $\hat{T}$ na neoznačenem besedilu lahko dobimo z uporabo Viterbijevega algoritma [55]. Napovedovanje s skritimi markovskimi modeli je hitro, vendar zaradi predpostavk ne deluje vedno najbolje. Slabo deluje na jezikih s kompleksno gramatiko, kjer so besedne vrste odvisne od več kot le ene predhodne besedne vrste.

Drugi pristop je označevanje z nadzorovanim učenjem. Kot razredno spremenljivko definiramo besedno vrsto, kot atribute pa različne lastnosti besede in njenih okoliških besed. Primeri dobrih atributov, ki jih lahko uporabimo, so:

- okoliške besede $(...w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2...})$,
- okoliške oznake besednih vrst $(...t_{i-2}, t_{i-1}, t_{i+1}, t_{i+2...})$,
- vse možne besedne vrste okoliških besed, pridobljene iz slovarja,
- predpone besede (prvih n črk besede za nekaj različnih vrednosti n),
- končnice besede (zadnjih n črk besede za nekaj različnih vrednosti n),
- ali beseda vsebuje številko,
- ali beseda vsebuje veliko črko,
- ali se beseda začne z veliko začetnico.

Na podlagi teh atributov lahko iz ročno označene učne množice zgradimo model z algoritmom nadzorovanega učenja. S tem lahko napovemo besedne vrste neoznačenega besedila. Modeli nadzorovanega učenja so bolj kompleksni od skritih markovskih modelov. Zaradi tega delujejo bolje na jezikih, pri katerih besedni red ni točno določen, kot je na primer slovenščina.

2.2.2 Semantična analiza

Na področje semantične analize spadajo metode, ki se ukvarjajo z razumevanjem pomena besedila. Primer take metode je razdvoumljanje pomena besed (word-sense disambiguation), kjer je cilj določiti pomen besede z več možnimi pomeni iz njenega konteksta. Za reševanje problemov semantične analize pogosto uporabimo metode strojnega učenja. Pri tem si pomagamo z zunanjimi podatkovnimi viri v obliki slovarjev, leksikonov in podatkovnih baz. Primer podatkovnega vira je podatkovna baza besed WordNet [15], kjer

so besede združene v skupine glede na njihov pomen. Poleg tega so skupine besed povezane glede na njihove medsebojne semantične odnose.

Eden izmed zanimivih problemov semantične analize je problem semantične podobnosti, kjer želimo za dve besedi, povedi ali besedili definirati mero, ki ustrezno napove ali imata besedi podoben pomen. To je težak problem, ker imajo lahko besede v različnih kontekstih povsem drugačen pomen. Iskanje semantične podobnosti dodatno otežijo besedilni konstrukti kot so sarkazem, metafore in ironija. Zaradi tega imajo lahko povedi, ki se na prvi pogled zdijo podobne, povsem drugačen pomen. Za iskanje semantične podobnosti je pogosto potrebno uporabiti metode strojnega učenja. Primera takih metod sta latentna semantična analiza (LSA) [33] in točkovna skupna informacija (pointwise mutual information ali PMI) [11].

V zadnjih letih se za probleme povezane s semantično podobnostjo pogosto uporabljano globoke nevronske mreže [49].

Vektorske vložitve besed

Pomembno področje semantične analize so vektorske vložitve besed (word embeddings). Z vektorsko vložitvijo besed želimo posamezne besede iz nekega besedila pretvoriti v vektorsko obliko tako, da dobljeni vektorji ustrezajo semantični podobnosti med besedami. Besede, ki so si med seboj semantično podobne, se preslikajo v vektorje, ki se nahajajo blizu drug drugega. Iz take predstavitve lahko razberemo pomensko povezavo med različnimi besedami.

Za izgradnjo vektorskih vložitev besed obstaja več različnih metod. Preprost način je metoda latentne semantične analize (LSA) [32]. Ta besede najprej predstavi z matriko besed in dokumentov, kjer vrstice predstavljajo besede, stolpci pa frekvence pojavitve besed v posameznih dokumentih korpusa. Metoda LSA to matriko razcepi z uporabo metode dekompozicije edinstvenih vrednosti (singular value decomposition ali SVD).

Naj bo X matrika besed in dokumentov dimenzije $|V| \times c$, kjer je $|V|$ število besed v vseh dokumentih in c število dokumentov. SVD matriko izrazi kot:

$$X = U\Sigma V^T,$$

kjer je U matrike dimenzije $|V| \times m$, Σ diagonalna matrika dimenzije $m \times m$ in V^T matrika dimenzije $m \times c$. m je rang matrike X. V matriki U vrstice predstavljajo besede v novem prostoru dimenzije m. V matriki V^T stolpci predstavljajo dokumente v novem prostoru dimenzije m. Σ je diagonalna matrika, kjer vrednosti na diagonali izražajo pomembnost posameznih dimenzij. S temi vrednostmi lahko iz U izberemo le k najpomembnejših stolpcev in dobimo matriko U_k dimenzije $|V| \times k$, katere stolpci so vektorske vložitve besed. Matrika $X_k = U_k \Sigma_k V_k^T$ je približek matrike X ranga k z najmanjšo napako.

Nekatere metode uporabljajo nevronske mreže. Primera takih metod sta vložitvi word2vec [36] in GloVe (global vectors for word representation) [40]. Metoda word2vec za izračun vektorskih vložitev besed uporabi dva različna jezikovna modela: zvezno vrečo besed (continuous bag-of-words ali CBOW) ali preskočne n-grame (skip-grams). Model CBOW predpostavi, da je verjetnost pojavitve posamezne besede w odvisna od njenega konteksta c. Kontekst definiramo kot n besed pred in za njo, pri čimer vrstni red okoliških besed ni pomemben. Pri izgradnji modela želimo izračunati verjetnosti $p(w|c)$. Preskočni n-grami imajo enako predpostavko, vendar želimo pri njih namesto verjetnosti $p(w|c)$ izračunati verjetnosti $p(c|w)$. To pomeni, da želimo za vsako besedo napovedati verjetnosti kontekstov, v katerih se beseda pojavi. Preskočni n-grami vrnejo boljše rezultate, vendar so počasnejši od zvezne vreče besed. Pri magistrskem delu smo zaradi tega uporabili metodo word2vec s preskočnimi n-grami.

Vektorsko vložitev besed s preskočnimi n-grami dobimo tako, da modelu dodamo parametre θ, in nato optimiziramo parametre θ tako, da maksimiziramo produkt verjetnosti glede na pojavitve besed v korpusu:

$$\underset{\theta}{\operatorname{argmax}} \prod_{(w,c) \in D} p(c|w;\theta),$$

kjer je D množica vseh parov (w, c), ki se pojavijo v korpusu. Word2vec

za parameterizacijo modela uporabi nevronsko mrežo z dvema nivojema. To pomeni, da lahko model $p(c|w;\theta)$ opišemo s soft-max funkcijo na sledeč način:

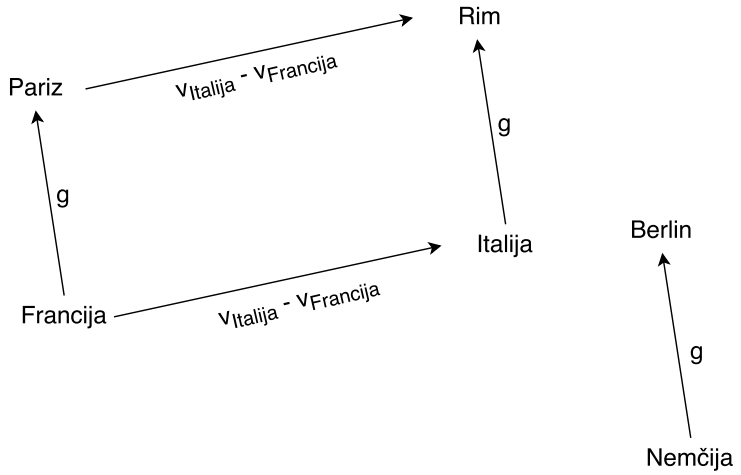
$$p(c|w;\theta) = \frac{e^{v_c \cdot v_w}}{\sum_{c' \in C} e^{v_{c'} \cdot v_w}},$$

kjer sta v_c in v_w vektorski predstavitvi konteksta c in besede w, C pa množica vseh kontekstov, ki se pojavijo v korpusu. Parametri θ so v_{ci} in v_{wi} za $c \in C$, $w \in V$, $i \in 1, ..., d$, kjer je V množica vseh besed v korpusu, C množica vseh kontekstov v besede v in d število komponent vsakega vektorja. Običajno se uporabi vrednost $d = 300$. Če celotno enačbo zapišemo v logaritemski obliki dobimo:

$$\underset{\theta}{\operatorname{argmax}} \sum_{(w,c) \in D} \log p(c|w) = \sum_{(w,c) \in D} (\log e^{v_c \cdot v_w} - \log \sum_{c'} e^{v_{c'} \cdot v_w})$$

Po optimizaciji zgornje enačbe dobimo matriko θ velikosti $|V| \cdot |C| \cdot |d|$. Vektorji v_w predstavljajo vektorske vložitve besed, ki smo jih želeli dobiti. Velja, da imajo besede, ki se pojavijo v podobnih kontekstih podobne vektorje v_w. Poleg tega dobljeni vektorji vsebujejo informacije o odvisnostih med besedami. Če na primer vzamemo besede *Francija*, *Pariz*, *Italija* in *Rim*, lahko metoda word2vec ustrezno prepozna, da sta Rim in Pariz glavni mesti Italije in Francije. V vektorskih vložitvah se to izrazi tako, da velja $v_{\text{Pariz}} - v_{\text{Francija}} + v_{\text{Italija}} = v_{\text{Rim}}$. Metoda word2vec se je zmožna naučiti poljubnih odvisnosti med besedami. Grafični prikaz odvisnosti med dobljenimi vektorji je prikazan na sliki 2.1.

Metoda GloVe deluje na podoben način. GloVe definira P_{ij} kot verjetnost, da se beseda i pojavi v kontekstu (okolici) besede j. Te verjetnosti izračunamo iz korpusa, na katerem učimo model. Nato definira model $F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$. w_i, w_j in $\tilde{w_k}$ so vektorske vložitve besed i, j in k. Model F je odvisen od različnih parametrov, podrobneje opisanih v originalnem članku [40]. Z optimizacijo napake lahko najdemo vektorje w_i, w_j in w_k, ki zagotovijo, da za koeficient $\frac{P_{ik}}{P_{jk}}$ veljajo sledeče lastnosti:



Slika 2.1: Grafični prikaz vektorjev dobljenih z vektorskimi vložitvami besed z metodo word2vec. Če vektorski vložitvi besede *Pariz* prištejemo vektor $v_{\text{Italija}} - v_{\text{Francija}}$ dobimo vektorsko vložitev besede *Rim*. Vektor g predstavlja odvisnost med državo in njenim glavnim mestom.

- Če se besede i, j in k pojavijo v skupnih kontekstih, mora biti koeficient blizu 1.
- Če se besedi i in k pojavita v skupnih kontekstih, j pa v drugačnih kontekstih, mora biti koeficient velik.
- Če se besedi j in k pojavita v skupnih kontekstih, i pa v drugačnih kontekstih, mora biti koeficient majhen.

2.3 Analiza omrežij

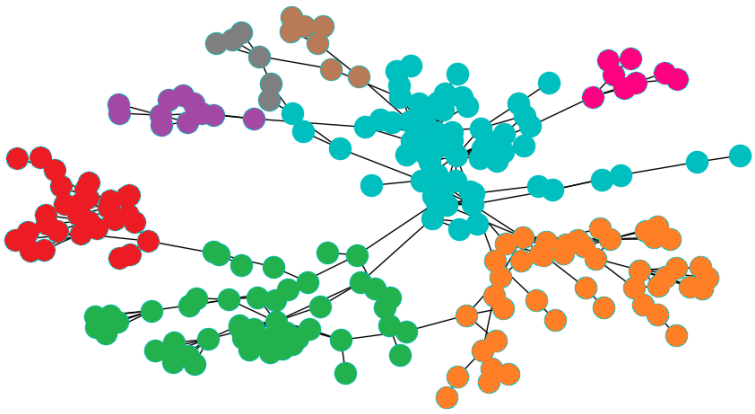
Pri tekstovnih bazah podatkov so poleg podatkov v tekstovni obliki pogosto prisotni tudi metapodatki v obliki omrežij. Primer so objave na družabnem omrežju Facebook, kjer imamo poleg besedila na voljo tudi podatke o avtorju in uporabnikih, ki so objavili komentar objave. Ker so uporabniki med sabo povezani glede na prijateljstva in skupine, ki jim pripadajo, lahko iz upo-

rabnikov zgradimo omrežje, ki pogosto vključuje koristne podatke. Omrežje lahko zgradimo tudi iz objav. Ker so podatki v obliki omrežij pogosti, obstaja veliko metod, ki poskušajo iz njih izluščiti koristne informacije. To je zapleteno, ker so koristne informacije v omrežjih lahko prisotne na različne načine.

Pogosta so tudi omrežja citatov [43]. To so omrežja, kjer vozlišča predstavljajo znanstveni članki. Članek c_i je povezan s člankom c_j, če c_j citira c_i. Ker članki citirajo sorodne članke, lahko iz takih omrežij dobimo koristne informacije o podobnosti člankov. Primer metode, ki to stori je kocitacija (co-citation) [53], ki definira indeks kocitacije med člankoma c_i in c_j kot število člankov, ki citirajo oba članka. Članki z večjim indeksom kocitacije so si verjetno med seboj bolj podobni kot tisti z manjšim. Sorodna mera je bibliografsko sklapljanje (bibliographic coupling) [28], kjer definiramo moč sklapljanja med člankoma c_i in c_j kot število citiranih člankov, ki so skupni člankoma c_i in c_j.

Zanimiv je tudi problem odkrivanja skupnosti (community detection) [16]. Pri njem predpostavimo, da so vozlišča omrežja združena v različne skupine, ki jim pravimo skupnosti. Za skupnost obstaja več različnih definicij, na primer skupnosti so tista podomrežja, kjer so vozlišča v podomrežju med seboj tesno povezana, podomrežje pa je šibko povezan s preostankom omrežja. Želimo najti postopek, ki skupnosti pravilno zazna. To je koristno, ker so si vozlišča znotraj skupnosti med seboj verjetno bolj podobna kot vozlišča iz različnih skupnosti. Slika 2.2 prikazuje, kako lahko graf razdelimo na skupnosti.

Pri omrežjih citatov lahko z zaznavanjem skupnosti poskusimo razdeliti članke na znanstvena področja. Članki iz iste skupnosti bolj verjetno spadajo v isto področje, kot članki iz različnih skupnosti, saj članki pogosteje citirajo članke iz istega raziskovalnega področja kot članke iz drugih področij. Podatke, ki jih pridobimo z zaznavanjem skupnosti lahko uporabimo v povezavi z drugimi metodami strojnega učenja in s tem izboljšamo rezultate.



Slika 2.2: Primer grafa, razdeljenega v skupnosti, ki so označene z različnimi barvami. Graf bi lahko razdelili tudi drugače, saj za skupnosti nimamo stroge definicije. To pomeni, da različni algoritmi zaznavanja skupnosti vrnejo različne rezultate.

2.4 Orodja za upravljanje konferenc

Organizacija znanstvenih konferenc je časovno zahtevno opravilo. Pred začetkom konference morajo organizatorji pregledati veliko število člankov, izbrati kateri bodo sprejeti na konferenco in sestaviti ustrezen urnik predstavitev člankov. Brez pomoči računalniških orodij bi organizacija večjih konferenc vzela zelo veliko časa. Organizatorji si organizacijo olajšajo z uporabo računalniških orodij, ki jim pri tem nudijo podporo.

Glavno opravilo, ki ga taka orodja poskušajo izboljšati, je recenzentski postopek, pri katerem so izbrani članki, ki bodo predstavljeni na konferenci. Recenzentski postopek poteka tako, da raziskovalci organizatorjem pošljejo članke, ki jih želijo objaviti na konferenci. Skupina recenzentov, ki sestavlja programski odbor konference, pregleda članke iz svojega raziskovalnega področja in izbere članke, ki bodo sprejeti. Računalniška orodja postopek poenostavijo tako, da raziskovalcem nudijo preprost način za pošiljanje člankov. Članke razporedijo recenzentom tako, da vsak dobi ustrezno število člankov iz svojega področja. Organizatorji glede na mnenja recenzentov določijo,

kateri članki so sprejeti in kateri ne. Primera pogosto uporabljenih orodij za upravljanje konferenc sta EasyChair [2] in OpenConf [6]. Ostala orodja za upravljanje konferenc so Microsoft Conference Management Toolkit [5], IAPR Commence Conference Management System [4] in EDAS: Editor's Assistant [3]. EasyChair uporabnikom ponuja sledeče funkcionalnosti:

- upravljanje in nadzor programskega odbora,
- izbira in nadzor člankov, do katerih lahko dostopajo člani odbora,
- samodejno oddajo člankov avtorjev,
- zajem preferenc recenzentov glede člankov,
- samodejno dodelitev člankov recenzentom glede na njihove želje,
- sistem za oddajo ocen člankov s strani recenzentov,
- pošiljanje in spremljanje spletne pošte članom programskega odbora, recenzentom in avtorjem člankov,
- spletno mesto za diskusijo o člankih,
- podporo odziva avtorjev na ocene recenzentov,
- izdelavo konferenčnega zbornika,
- ustvarjanje programa konference,
- ustvarjanje brošur s programom konference.

OpenConf ponuja uporabnikom podobne funkcionalnosti.

2.5 Podobne raziskave

Samodejno razvrščanje člankov v urnik konference še ni raziskano. Člankov, ki opišejo celoten postopek razvrščanja, nam ni uspelo najti. Nekateri članki

opišejo metode za gručenje z omejitvami velikosti gruč, kot sta CSCLP (Clustering Algorithm with Size Constraints and Linear Programming) [54] in K-MedoidsSC (K-Medoids algorithm with Size Constraints) [54]. V [27] je predstavljena metoda, ki za gručenje z omejitvami uporabi hierarhično gručenje.

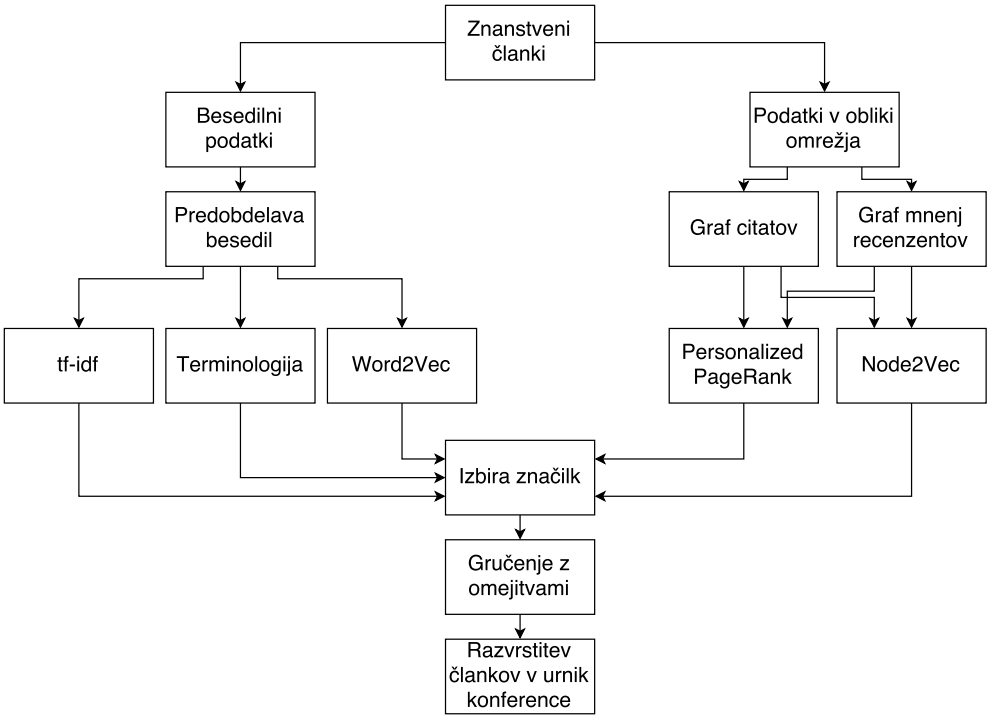
Pomemben del našega dela je semantična analiza in iskanje podobnih znanstvenih člankov, kar je prisotno pri različnih problemih. Konferenčni priporočilni sistemi [57] na podlagi podobnosti med članki udeležencem konferenc priporočijo predstavitve člankov, ki bi jih verjetno zanimale. Podobnosti med članki definirajo z uporabo analize omrežij na podoben način, kot to storimo v našem delu. Primer uporab metod obdelave naravnega jezika prilagojenih znanstvenim člankom je orodje Dr Inventor [47], ki lahko iz znanstvenih člankov izlušči koristne informacije, kot so reference, in zgradi povzetek članka.

Poglavje 3

Samodejno razvrščanje člankov v urnik konference

Pri implementaciji metode za samodejno razvrščanje člankov v urnik konference smo uporabili kombinacijo metod z različnih področij strojnega učenja. Najprej smo z različnimi metodami obdelave naravnega jezika izluščili besedilne podatke, ki najbolje opišejo vsebino članka. Poleg besedilnih podatkov smo uporabili podatke v obliki omrežij. Za luščenje teh podatkov smo uporabili metode s področja analize omrežij. Dobljene podatke smo omejili z uporabo izbire značilk. Z gručenjem smo združili podobne članke v gruče, ki ustrezajo sejam vnaprej definiranega urnika. Ker standardne metode gručenja ne dovoljujejo, da vnaprej definiramo velikosti gruč, smo prilagodili metodo k-voditeljev tako, da ta zagotovi, da dobljene gruče ustrezajo strukturi urnika. Metodo smo implementirali v sklopu spletne aplikacije, ki organizatorjem konferenc omogoča samodejno razvrščanje člankov v urnik konference preko spletnega vmesnika. Diagram poteka, ki prikazuje uporabljene metode, je predstavljen na sliki 3.1.

V tem poglavju opišemo metode, ki smo jih uporabili za samodejno razvrščanje člankov v urnik konference. V poglavju 3.1 predstavimo, kako smo članke v obliki pdf dokumentov pretvorili v tekstovno obliko. Predstavimo predobdelavo dobljenih besedil, ki smo jo izvedli za boljše delovanje



Slika 3.1: Diagram postopka za samodejno razvrščanju člankov v urnik konference.

naslednjih korakov. V poglavju 3.2 predstavimo metode za pridobivanje značilk, ki smo jih uporabili za iskanje podobnih člankov. V poglavju 3.3 opišemo algoritem za gručenje z omejitvami, ki s pomočjo značilk članke razporedi v vnaprej definirano strukturo urnika.

3.1 Predobdelava besedil

Konferenčni članki so objavljeni kot binarni dokumenti v formatu PDF (Portable Document Format) ali v formatu Microsoft Word. Te oblike niso primerne za metode obdelave naravnega jezika, ker je pri njih težko dostopati do besedila. Zato članke najprej pretvorimo v tekstovno obliko.

Omejili smo se na članke objavljene v obliki PDF, saj je ta najpogostejša pri člankih s področja računalništva in informatike. Za pretvorbo dokumen-

tov v obliki PDF v tekstovno obliko obstaja več prosto dostopnih knjižnic in programov. Uporabili smo orodje pdftotext, ki je na operacijskih sistemih Windows na voljo skupaj s programom Xpdf [7].

Po pretvorbi člankov v tekstovno obliko smo nad njimi izvedli različne metode predobdelave besedil, ki izboljšajo delovanje metod obdelave naravnega jezika. Začeli smo z odstranjevanjem delov besedil, ki ne nosijo podatkov o pomenu članka. Odstranili smo podatke o založniku in avtorjih, ki se pojavijo pred začetkom povzetka, in reference, ki smo jih obravnavali posebej, kot je opisano v poglavju 3.2.4. Odstranili smo besede, ki vsebujejo številke. Te besede se skoraj vedno pojavijo v enačbah in niso koristne za iskanje podobnih člankov. Primer so spremenljivke $x_1, x_2, ...x_n$, ki se pogosto pojavijo v enačbah ali opisih algoritmov. Izven enačb se številke pojavijo pri citatih v besedilu. Ker citate obravnavamo posebej tudi take primere odstranimo.

Z uporabo slovarja nepotrebnih besed (stop words) smo odstranili pogosto uporabljene besede, ki ne vsebujejo podatkov o pomenu besedila. Primer takih besed so vezniki, časovna določila, pogosti glagoli, samostalniki in predlogi. Odstranili smo naglasna znamenja, kot sta ostrivec (á) in krativec (à). Z naglasnimi znamenji je možno isto besedo zapisati na več različnih načinov, kar oteži semantično analizo besedila.

Kot zadnji korak predobdelave smo izvedli tokenizacijo, ki besedilo razdeli na posamezne besede. To je lahko v določenih tipih besedil, na primer v slovnično nepravilnih objavah na družabnih omrežjih, težko opravilo. V našem primeru so slovnične napake redke, zato smo uporabili preprosto tokenizacijo, ki besede loči glede na presledke, vezaje, pike, vejice in druge podobne znake.

3.2 Metode za ekstrakcijo značilk

Po predobdelavi in tokenizaciji besedila niso več predstavljena v tekstovni obliki, ampak s seznami besed, ki se v njih pojavijo. Ta oblika ni primerna za iskanje podobnih člankov. Še vedno je namreč prisotno veliko število besed,

ki ne nosijo koristnih informacij. Pomemben je vrstni red besed, kar ni dobro. Če zamenjamo dve povedi se bo predstavitev besedila spremenila, tudi če se pomen ni. Zaradi tega bi radi besedila pretvorili v obliko, primernejšo za iskanje podobnih člankov.

Način, kako lahko to storimo, je ekstrakcija značilk. Namesto, da besedilo predstavimo z besedami, iz njega izluščimo ključne lastnosti ali značilke. V nadaljnjih korakih so besedila tako predstavljena v obliki, ki vključuje samo koristne podatke. Želimo, da so značilke neodvisne druga od druge, kar pomeni, da vrstni red značilk ne bi vplival na končni rezultat. Ekstrakcija značilk je pogosta v skoraj vseh oblikah strojnega učenja. Zanjo smo uporabili metode obdelave naravnega jezika v kombinaciji z metodami analize grafov.

3.2.1 Vektorske predstavitve besedil

Značilke besedil izluščimo z namenom vektorske predstavitve besedil. Celotno besedilo želimo pretvoriti v vektor, ki ohrani podatke o frekvencah prisotnih besed.

Preprosta metoda za predstavitev besedil je predstavitev vreča besed (bag-of-words). Pri tej predstavitvi preštejemo pojavitve posameznih besed v besedilu in jih shranimo v vektor. Če imamo povedi ,,*Žena sedi na stolu in šiva*“, ,,*Mož sedi na stolu in gleda televizijo*“ in ,,*Riba plava v morju*“ bi s predstavitvijo z vrečo besed dobili vektorje, prikazane v tabeli 3.1.

Predstavitev z vrečo besed deluje dobro, če želimo dokumente razvrstiti

Tabela 3.1: Predstavitev z vrečo besed za stavke ,,*Žena sedi na stolu in šiva*“, ,,*Mož sedi na stolu in gleda televizijo*“ in ,,*Riba plava v morju*“.

mož	žena	sedi	na	stolu	in	šiva	gleda	televizijo	riba	plava	v	morju
0	1	1	1	1	1	1	0	0	0	0	0	0
1	0	1	1	1	1	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	1	1	1	1

v skupine, ki si delijo majhno število besed. V tem primeru bosta vektorja dokumentov iz različnih skupin skoraj pravokotna drug na drugega, medtem ko bosta vektorja dokumentov iz iste skupine imela skoraj enako smer. V primeru iz tabele 3.1 si prva dva vektorja delita večino besed in imata podobno vsebino, tretji pa ima različno vsebino in je pravokoten nanju. V praksi so taki primeri redki. Obstaja mnogo pogostih besed, ki so z visoko frekvenco prisotne v večini dokumentov, ne glede na njihovo vsebino. Primer so besede *je*, *v*, *in*, *na*, *da*, *se*. Zaradi tega bodo predstavitve različnih dokumentov z vrečo besed bolj podobne, kot bi si želeli. Take besede pogosto izločimo že v postopku predprocesiranja besedil, vendar je težko izločiti vse.

Predstavitev z vrečo besed lahko izboljšamo tako, da zmanjšamo frekvence besed, ki se pogosto pojavijo v vseh dokumentih. Radi bi utežili člene vektorja tako, da bi imele pomembne besede visoko težo, nepomembne besede pa nizko. Takšno oteževanje je tf-idf (term frequency–inverse document frequency) [44], ki za vsako besedo izračuna dva faktorja. Prvi je frekvenca besed (term frequency, $\mathrm{tf}(t, d)$), ki meri kolikokrat se beseda t pojavi v dokumentu d. Drugi je inverzna dokumentna frekvenca (inverse document frequency $\mathrm{idf}(t)$), ki jo izračunamo kot:

$$\mathrm{idf}(t) = \frac{N}{n_t},$$

kjer je N število vseh dokumentov in n_t število dokumentov, ki vsebujejo besedo t. Besede, ki se pojavijo v vseh dokumentih bodo imele $\mathrm{idf} = 0$, tiste, ki se pojavijo v majhnem številu dokumentov, pa bodo imele visok idf. To je koristno, ker so besede, ki se pojavijo v velikem številu dokumentov manj informativne za razlikovanje dokumentov, kot tiste, ki se pojavijo v majhnem številu dokumentov.

Končno mero $\text{tf-idf}(t, d)$ dobimo kot:

$$\text{tf-idf}(t, d) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t).$$

Kot osnovne značilke člankov smo uporabili matrike tf-idf vektorjev dokumentov. Metoda tf-idf lahko vrne dobre rezultate. Uporabimo jo kot

izhodiščno metodo, s katero lahko primerjamo naprednejše metode.

3.2.2 Luščenje terminologije

Z ekstrakcijo značilk želimo iz besedil izluščiti tiste podatke, ki nam najbolj pomagajo najti pomensko podobne članke. Želimo torej izluščiti podatke o pomenu besedila. To lahko storimo tako, da v celotnem besedilu poiščemo besede, ki najbolje opišejo njegov pomen. Pri tehničnih besedilih, kot so znanstveni članki, so dobra izbira besede iz strokovne terminologije. Na primer, besedila s področja nadzorovanega učenja bodo vsebovala mnogo besed iz terminologije nadzorovanega učenja, medtem ko bodo besedila s področja optimizacije vsebovala le malo takih besed. Besede strokovne terminologije najdemo z metodami luščenja terminologije.

Pri večini metod luščenja terminologije si lahko pomagamo z zunanjim korpusom, ki je sestavljen iz tehničnih besedil z istega področja kot so besedila, iz katerih želimo izluščiti terminologijo. Korpusi so koristni, ker vsebujejo mnogo večje število besedil, kot besedila iz katerih želimo izluščiti terminologijo. Kot korpus smo uporabili članke s področja strojnega učenja, saj na to področje spadajo članki iz podatkovne množice, ki smo jo uporabili za testiranje.

Pri luščenju terminologije želimo vsaki besedi pripisati številčno oceno tako, da imajo besede, ki bolj verjetno spadajo med strokovno terminologijo, visoko oceno. Ocene izračunamo s kombinacijo različnih lastnosti besede, kot so besedna vrsta, okoliške besede, število pojavitev besede v celotnem korpusu in število dokumentov, kjer se beseda pojavi. Izračun ocene opravimo v dveh fazah. V prvi z uporabo hitrih metod poiščemo kandidate za terminološke fraze, saj je terminoloških fraz v besedilih skoraj vedno manj kot splošnih besed. Zato najprej hitro izločimo besede, ki zelo verjetno ne spadajo pod terminološke fraze. V drugi fazi podrobneje pregledamo kandidate in izberemo tiste, ki so dejansko terminološke fraze.

V našem delu smo v prvi fazi izločili besedne n-grame, ki niso ustrezali določenim besednim oblikam. Predpostavili smo, da bo večina terminoloških

Tabela 3.2: Oblike terminoloških fraz pri iskanju kandidatov.

Oblika fraze	Primer terminološke fraze
Samostalnik	Kernel
Samostalnik, samostalnik	Machine learning
Pridevnik, samostalnik	Neural network
Pridevnik, pridevnik, samostalnik	Deep neural network
Pridevnik, samostalnik, samostalnik	Deep learning theory
Samostalnik, samostalnik, samostalnik	Column subset selection

fraz sestavljenih samo iz samostalnikov ali iz kombinacije samostalnikov in pridevnikov. To je pogost pristop, saj večina terminološke fraz ustreza taki obliki. V tabeli 3.2 so prikazane uporabljene oblike in primeri fraz, ki ustrezajo tem oblikam.

V drugi fazi smo najprej odstranili kandidate, ki zelo verjetno niso terminološke fraze. Odstranili smo sledeče kandidate:

Kandidati, ki so se pojavili v manj kot treh člankih - Ekstrakcijo terminologije želimo uporabiti za gručenje podobnih člankov v gruče. Te gruče želimo uporabiti za razvrščanje člankov v seje urnika konference, ki večinoma vsebujejo vsaj tri članke, zato ne želimo manjših gruč. Tudi, če so taki kandidati terminološke fraze, nam pri gručenju ne bodo pomagali, zato jih odstranimo.

Kandidati, ki se pojavijo v več kot 15% člankov - Ne želimo prevelikih gruč, zato odstranimo kandidate, ki se pojavijo v velikem številu različnih člankov. S tem izločimo terminološke fraze, ki so vezane na celotno področje konference in ne na podpodročja, ki bi jih z gručenjem radi odkrili.

Kandidati s pogostimi pridevniki - Zaradi oblike kandidatov terminoloških fraz, predstavljenih v tabeli 3.2, se lahko zgodi, da imamo med kandidati samostalnik, ki je terminološka fraza, in kandidata, ki pred samostalnikom vsebuje še dodaten pridevnik, ki ne spada pod izraz iz

terminologije. Na primer najdemo *kernel* in *different kernel*. Temu se izognemo tako, da odstranimo fraze s pogostimi pridevniki. V našem primeru odstranimo *different kernel* in obdržimo *kernel*. Iz seznama pogostih pridevnikov smo odstranili pridevnike, ki se pogosto pojavijo v terminologiji računalništva in informatike, kot so *deep*, ki se pojavi pri globokem učenju, in *big*, ki se pojavi pri izrazu *big data*.

Kandidati krajši od pet črk - Takšni kandidati se skoraj vedno pojavijo zaradi napak pri pretvorbi pdf dokumentov v tekstovno obliko, ali pa so simboli iz enačb in nas ne zanimajo.

Kandidati, ki ustrezajo imenom zbornikov - Članki velikokrat vsebujejo ime zbornika konference, v katerem so objavljeni. Imena zbornikov spadajo pod terminološke fraze, vendar niso koristna za iskanje podobnih člankov, zato jih odstranimo.

Kandidati, kjer posamezna beseda vsebuje le eno črko - To so skoraj vedno kandidati oblike *Graph G* ali *Matrix M*, ki se pojavijo v definicijah. Taki kandidati nam ne povejo veliko o pomenu besedila, zato jih izpustimo.

Vsakemu kandidatu priredimo oceno, ki ustreza verjetnosti, da je kandidat terminološka fraza. Za dobre terminološke fraze morata veljati dve lastnosti:

Fraza se pogosto pojavi v dokumentu, ki jo vsebuje. Zanimajo nas predvsem fraze, ki dobro opišejo pomen besedila. Pogoste fraze verjetneje odražajo pomen besedila kot redke.

Fraza se pojavi v dovolj velikem številu dokumentov. Želimo najti splošno uporabljene terminološke fraze. To pomeni, da se morajo fraze pojaviti v dovolj velikem številu dokumentov. Redke fraze so preveč specifične.

Če imamo na voljo korpus s splošnim besediščem, lahko uporabimo še sledečo lastnost:

Tabela 3.3: Primeri kandidatov za terminološke izraze, ki se pojavijo v korpusu s splošnim besediščem.

Več kot 100 pojavitev	0 do 100 pojavitev	0 pojavitev
information networks	many contexts	proximal map
detection system	additional challenges	scale-free distribution
fast pace	same row	optimistic policy
natural conditions	rate hop	optimal algorithm
other goals	real student	incoherence requirement
	principle components	normalized weighing

Fraza se mora redko pojaviti v splošnem korpusu. Iščemo terminološke izraze, specifične za neko tehnično področje. Takšni izrazi se izven področja redko pojavijo.

V našem primeru se je izkazalo, da se večbesedni kandidati za terminološke izraze skoraj nikoli ne pojavijo v splošnem besedišču. Za korpus s splošnim besediščem smo uporabili *Corpus of Contemporary American English (COCA)* [12]. Korpus vsebuje več kot 520 milijonov angleških besed iz televizijskih in radijskih programov, literarnih del, revij, časopisov in znanstvenih člankov. Je eden največjih korpusov v angleškem jeziku. Tabela 3.3 prikazuje nekatere kandidate za terminološke izraze, ki so se pogosto pojavili v splošnem korpusu in nekatere, ki se niso. Razvidno je, da lahko z uporabo splošnega korpusa odstranimo nekatere kandidate, ki niso terminološki izrazi, ampak spadajo v splošno besedišče. Primer sta kandidata *additional challenges* in *fast pace*. Nekateri kandidati, ki se pojavijo v splošnem besedišču, kot na primer *principle components*, so kljub temu terminološki izrazi, vendar so taki kandidati redki.

Ocene kandidatov pridobimo s kombinacijo opisanih lastnosti. Vsakega kandidata za terminološko frazo opišemo s tremi lastnostmi:

- Relativna frekvenca pojavitve kandidata v znanstvenih člankih, $F_s(t)$;
- Relativna frekvenco pojavitve kandidate v korpusu COCA, $F_g(t)$;

- Številom člankov, v katerih se kandidat pojavi, $D(t)$.

Končno oceno kandidata definiramo kot v [39]:

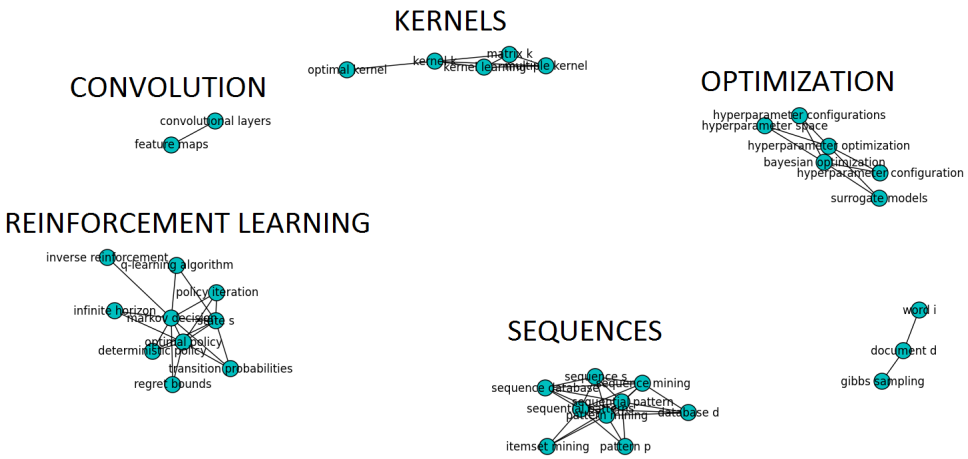
$$S(t) = 1 - \frac{1}{log_2(2 + \frac{F_s(t)D(t)}{F_g(t)})}$$

Da se izognemo deljenju z 0 pri kandidatih, ki se ne pojavijo v korpusu COCA, vrednostim $F_g(t)$ prištejemo majhno vrednost. Tabela 3.4 prikazuje najboljše in najslabše ocenjene kandidate za terminološke izraze, ki smo jih izluščili iz naše podatkovne baze. Slabo ocenjeni so izrazi, ki se pogosto pojavijo v splošnem besedišču (*insightful discussions*), tisti, ki se pojavijo v premajhnem številu člankov (*nsf iis-*) in tisti, ki se pojavijo v velikem številu člankov, vendar z majhno frekvenco (*d-dimensional vector*). Dobro ocenjeni kandidati so dejanski terminološki izrazi s področja strojnega učenja.

Tabela 3.4: Primeri nekaterih najboljših in najslabših bigramskih kandidatov za terminološke izraze.

Terminološki izraz	ocena $S(t)$		
Najboljši:		Najslabši:	
variational inference	0.15602	d-dimensional vector	0.00048
gaussian process	0.11889	nsf iis-	0.00048
step size	0.10156	learning community	0.00048
hinge loss	0.08178	d-dimensional feature	0.00047
logistic regression	0.08152	decision boundary	0.00047
graphical models	0.06531	discrete values	0.00046
graphical model	0.06095	model formulation	0.00046
		insightful discussions	0.00046

Iz izluščene terminologije smo zgradili graf terminologije. V njem sta posamezna terminološka izraza povezana, če se dovolj pogosto pojavita v istem članku. S tem določimo sorodnost različnih terminoloških izrazov in vizualiziramo terminologijo, uporabljeno v različnih člankih. Slika 3.2 predstavlja vizualizacijo najpogosteje uporabljenih terminoloških izrazov v delu podatkovne baze, ki smo jo uporabili za testiranje. Na sliki so prisotne gruče



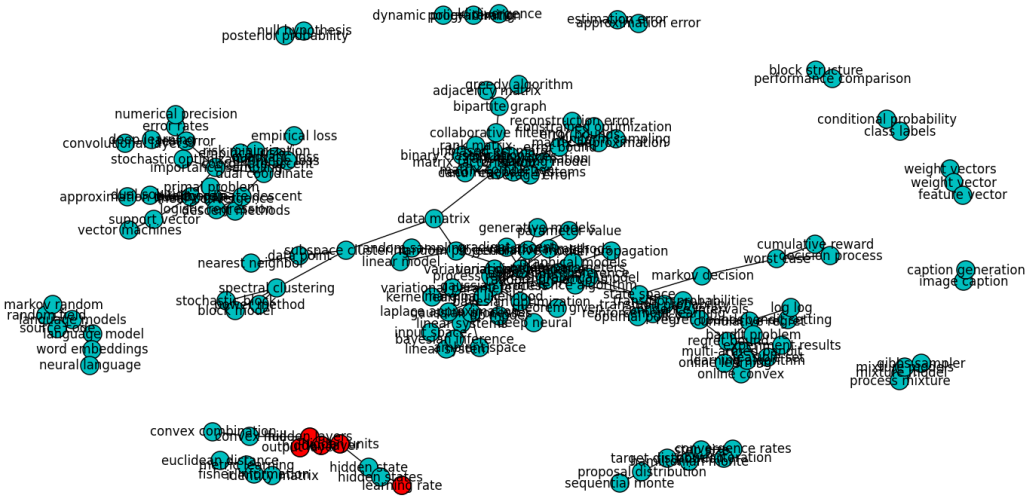
Slika 3.2: Vizualizacija grafa terminologije. Zaradi preglednosti so prikazani le najpogostejši terminološki izrazi. Vsaka gruča je označena s podpodročjem strojnega učenja, na katerega spada. Oznake predstavljajo podpodročje člankov, v katerih se pripadajoči izrazi najpogosteje pojavijo.

terminoloških izrazov, ki so tesno povezani z različnimi podpodročji strojnega učenja.

V dobljenem grafu želimo, da so terminološki izrazi iz posameznega članka čim bližje drug drugemu. Na podlagi tega smo optimizirali parametre pri luščenju terminologije tako, da smo minimizirali razdaljo med terminološkimi izrazi posameznih člankov. Vizualizacija terminoloških izrazov v posameznem članku je prikazana na sliki 3.3. V prikazanem primeru se terminološki izrazi članka nahajajo blizu drug drugemu, kar pomeni, da v je luščenje terminologije dobro delovalo.

3.2.3 Word2Vec

Vektorske predstavitve besedil, kot so matrika tf-idf in model vreče besed, ne upoštevajo podobnosti in odvisnosti med posameznimi besedami. Besedi



Slika 3.3: Vizualizacija terminoloških izrazov posameznega članka. Terminološki izrazi, ki se pojavijo v članku, so obarvani z rdečo barvo. Zaradi preglednosti je omejeno število terminoloških izrazov v grafu in število izrazov, ki so se pojavili v članku.

„*kralj*“ in „*kraljica*“ sta v modelu vreče besed obravnavani kot povsem ločeni besedi, čeprav se razlikujeta le v spolu. Pri semantični analizi bi radi zajeli odvisnosti med takimi besedami. To lahko storimo z vektorskimi vložitvami besed, ki preslikajo posamezne besede v vektorsko obliko tako, da imajo besede, ki so med seboj pomensko povezane, podobne vektorje. Za izgradnjo vektorskih vložitev lahko uporabimo več različnih metod, kot so LSA (latent semantic analysis) [13], GloVe (global vectors for word representation) [40] in word2vec [36]. Uporabili smo metodo word2vec, ki je ena izmed najpogosteje uporabljenih vektorskih vložitev besed. Vektorsko vložitev zgradimo glede na okolico posameznih besed (n besed pred in za besedo, ki ji želimo zgraditi vektorsko vložitev). Z uporabo nevronske mreže zgradimo napovedni model, ki napove besedo glede na podano okolico, ali okolico glede na podano besedo. Parametri zgrajenega modela delujejo kot vektorska vložitev besed.

Metoda word2vec sama po sebi ni primerna za iskanje podobnih člankov. Vrne namreč le vektorske vložitve besed v člankih in ne vektorske predstavitve celotnih člankov. Za iskanje podobnih člankov moramo vektorske predstavitve posameznih besed združiti v vektorsko predstavitev celotnega članka. To smo storili na dva načina:

S povprečjem vektorjev besed. Preprost način, da dobimo vektorsko predstavitev celotnega članka je, da izračunamo povprečje vektorskih predstavitev besed v članku. Tak pristop ni najboljši, saj zanemari informacije o položaju posameznih besed.

Z metodo doc2vec [34]. Metoda doc2vec preslika vektorske vložitve besed v vektorske predstavitve celotnega besedila z uporabo nevronske mreže. V večini primerov doseže boljše rezultate kot povprečje vektorjev, vendar je za njen izračun potrebujemo več časa.

Metoda doc2vec deluje podobno kot word2vec. Namesto, da iz konteksta napovemo besedo želimo zgraditi model, ki iz vektorja besedila in vektorjev nekaterih besed v besedilu napove naslednjo besedo besedila. Kot pri metodi word2vec za učenje modela uporabimo nevronsko mrežo. Za vhodni vektor

vzamemo združene vektorje besedila in besed, kot izhodni vektor pa besedo, ki jo želimo napovedati. Z zgrajenim modelom lahko izračunamo vektorje besedil za nova besedila.

3.2.4 Značilke iz omrežij

Znanstveni članki ne vsebujejo koristnih informacij le v besedilu. Članke lahko med seboj povežemo na različne načine, s čimer dobimo različna omrežja, ki lahko vsebujejo koristne informacije. Pri akademskih člankih najpogosteje uporabimo omrežje citatov, kjer vozlišča ustrezajo člankom. Vozlišče v_i je povezano z vozliščem v_j, če članek i citira članek j. V takšnih omrežjih je vsak članek blizu člankom, ki jih citira in ki so mu verjetno pomensko podobni.

Za znanstvene konference taka omrežja niso najboljša izbira. Objavljeni članki velikokrat pred tem niso bili dostopni, zaradi česar članki predstavljeni na isti konferenci redko citirajo drug drugega. Namesto omrežja citatov zato uporabimo omrežje bibliografskega sklapljanja. V takem omrežju sta vozlišči v_i in v_j med seboj povezani, če članka i in j vsebujeta vsaj eno skupno referenco. Ker so si pari člankov, ki si med seboj delijo večje število referenc, verjetno bolj podobni, povezave otežimo s številom referenc, ki si jih članka i in j delita.

Drugi tip omrežja, ki ga uporabimo, je omrežje mnenj recenzentov. Pred začetkom konference tipično poteka recenzentski postopek, med katerim so izbrani članki, ki bodo predstavljeni na konferenci. Za vsak članek skupina recenzentov presodi, ali je članek primeren za objavo. Ker so na konferencah lahko predstavljeni članki z različnih znanstvenih podpodročij, pred začetkom recenzentskega postopka vsak recenzent izrazi mnenje o tem, katere članke želi in katerih ne želi oceniti. Ta mnenja so lahko koristna pri iskanju podobnih člankov, saj recenzenti večinoma želijo oceniti članke, ki spadajo na njihovo raziskovalno področje. Iz mnenj recenzentov omrežje zgradimo tako, da povežemo dva članka, če je oba želel oceniti isti recenzent.

Če želimo informacije prisotne iz omrežij uporabiti v algoritmih strojnega

učenja, jih moramo najprej pretvoriti v vektorsko obliko, ki ohrani informacije o omrežju. Za to smo uporabili dve metodi: Personalized PageRank [38] in node2vec [21].

Personalized PageRank

Metoda Personalized PageRank (P-PR) je bila originalno zasnovana za uporabo na omrežjih spletnih strani. Za poljubno spletno stran izračuna, kako blizu so ji ostale spletne strani. Za to uporabi model naključnega sprehoda, ki predpostavi, da uporabnik dostopa do spletnih strani tako, da iz začetne strani ali naključno klika na hiperpovezave, ali pa se z majhno verjetnostjo vrne na začetno stran. Metoda Personalized PageRank za nabor spletnih strani izračuna verjetnosti, da uporabnik iz začetne strani pride do teh spletnih strani. S tem lahko vsaki spletni strani priredimo vektor verjetnosti, kjer imajo bližnje spletne strani višje verjetnosti od oddaljenih. Metodo lahko uporabimo na omrežju bibliografskega sklapljanja. Vsakemu članku priredimo vektor, ki opiše njegov položaj v grafu.

PageRank $R'(u)$ je definiran z enačbo

$$R'(u) = c \sum_{v \in B_u} \frac{R'(v)}{N_v} + cE(u),$$

kjer je u članek, ki mu želimo izračunati P-PR vrednost, B_u množica vseh člankov, ki s z njim povezani v omrežju, N_v število izhodnih povezav iz v in c faktor normalizacije, ki poskrbi, da je L_1 norma R' enaka 1. $E(u)$ je vir PageRank vrednosti, ki ima različne definicije glede na različico metode. Pri metodi P-PR je $E(u)$ vektor s toliko elementi, kot imamo člankov in sledečimi vrednostmi:

$$E(u) = \begin{cases} 1 & : \text{u je začetni članek} \\ 0 & : \text{drugače} \end{cases}$$

Če želimo personalized PageRank izračunati za 3 članke a, b, c, bi uporabili vektorje $E(a) = [1, 0, 0]^T$, $E(b) = [0, 1, 0]^T$, $E(c) = [0, 0, 1]^T$. Personalized PageRank računamo iterativno. Najprej nastavimo vrednosti $R'(u)'$ na

1 in nato iterativno računamo nove vrednosti po zgornji enačbi. Postopek končamo po vnaprej določenem številu iteracij.

Node2vec

Personalized PageRank je preprosta metoda, ki za pretvorbo omrežja v vektorsko obliko uporabi predvsem razdalje med vozlišči. Omrežja lahko koristne informacije hranijo tudi v drugačnih oblikah, ki jih personalized PageRank ne zazna. V omrežjih so velikokrat prisotne različne strukture, ki bi jih radi prepoznali, saj lahko s tem iz omrežja izluščimo koristne informacije.

Node2vec pretvori omrežje v vektorsko obliko z delno nadzorovanim učenjem. Temelji na modelu preskočnih n-gramov, ki je uporabljen tudi pri metodi word2vec. Pri metodi word2vec uporabimo preskočne n-grame na besedilih tako, da zgradimo model, ki iz posamezne besede napove najverjetnejši kontekst te besede (množico n besed pred in za besedo). Preskočne n-grame uporabimo na omrežjih tako, da model za poljubno vozlišče napove najverjetnejšo okolico vozlišča. Naj bo $G = (V, E)$ neko omrežje. Metoda node2vec definira pretvorbo vozlišča v vektorsko obliko s funkcijo $f : V \to \mathbb{R}^d$, kjer je d število dimenzij dobljenih vektorjev. Želimo maksimizirati verjetnost, da iz vektorske predstavitve vozlišča $f(u)$ napovemo okolico vozlišča u, ki jo označimo z $N_s(u)$:

$$\underset{f}{\operatorname{argmax}} \sum_{u \in V} p(N_s(u)|f(u))$$

Predpostavimo, da je verjetnost opažanja enega vozlišča iz okolice neodvisna od opažanja ostalih vozlišč, torej lahko zapišemo:

$$p(N_s(u)|f(u)) = \prod_{n_i \in N_s(u)} p(n_i|f(u))$$

Kot pri metodi word2vec uporabimo za definicijo verjetnosti softmax funkcijo:

$$p(n_i|f(u)) = \frac{e^{f(n_i) \cdot f(u)}}{\sum_{v \in V} e^{f(v) \cdot f(u)}}$$

Dobimo podobno optimizacijsko funkcijo kot pri metodi word2vec:

$$\max_f \sum_{u \in V} \Big[-\log \sum_{v \in V} e^{f(u) \cdot f(v)} + \sum_{n_i \in N_s(u)} f(n_i) \cdot f(u) \Big]$$

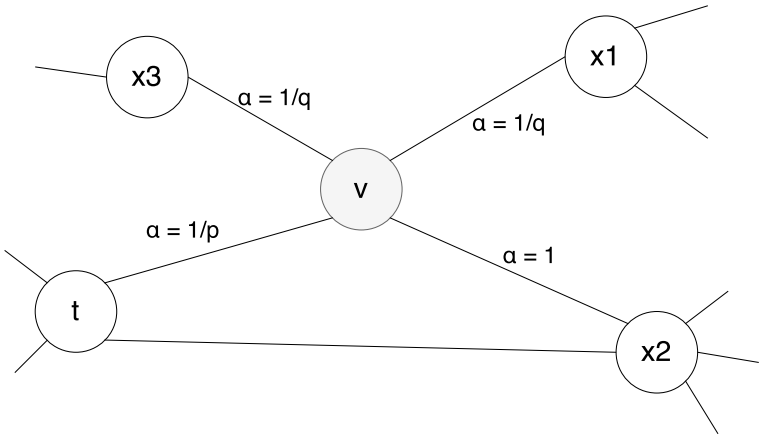
Ključen problem, ki ga moramo rešiti, če želimo metodo word2vec prenesti na omrežja, je definicija okolice vozlišča $N(u)$. Metoda node2vec uporabi naključne sprehode, ki so podobni tistim uporabljenim v metodi P-PR. Naj bo c_i i-to vozlišče v sprehodu, ki se začne z $c_0 = u$. Vozlišča v sprehodu so generirana z enačbo:

$$p(c_i = x | c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & : (v, x) \in E \\ 0 & : \text{drugače} \end{cases}$$

kjer je π_{vx} nenormalizirana verjetnost prehoda med vozliščema v in x; Z je normalizacijska konstanta, ki poskrbi da se verjetnosti prehodov seštejejo v 1. Verjetnost prehodov π_{vx} je definirana s pomočjo koeficientov p in q, s katerimi lahko vplivamo na delovanje naključnih sprehodov. Verjetnosti definiramo kot $\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$, pri čemer je v trenutno vozlišče sprehoda, t prejšnje vozlišče sprehoda in x naslednje vozlišče sprehoda. w_{vx} je pri oteženih omrežjih utež povezave (v, x), pri neoteženih omrežjih pa je $w_{vx} = 1$. $\alpha_{pq}(t, x)$ je pristranskost sprehoda, definirana kot:

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & : d_{tx} = 0 \\ 1 & : d_{tx} = 1 \\ \frac{1}{q} & : d_{tx} = 2 \end{cases}$$

kjer d_{tx} predstavlja neoteženo razdaljo med vozliščema t in x v omrežju. Grafični prikaz pristranskosti je prikazan na sliki 3.4. Pristranskost omrežja $\alpha_{pq}(t, x)$ nam omogoča, da pri uporabi metode node2vec vplivamo na definicijo okolice. Parameter q vpliva na strategijo iskanja. Če velja $q > 1$, bo naključni sprehod raje obiskal vozlišča blizu začetnega vozlišča t. Takšni sprehodi generirajo podobna vozlišča, kot jih generira iskanje v širino. Če velja $q < 1$ sprehod raje obišče vozlišča, ki so bolj oddaljena od t in je podoben iskanju v globino. Parameter p vpliva na velikost okolice. Če velja



Slika 3.4: Grafični prikaz pristranskosti naključnega sprehoda pri metodi node2vec. v je trenutno vozlišče sprehoda, t pa predhodno vozlišče sprehoda. Povezava do t ima pristranskost $\alpha_{pq}(t, x) = 1/p$. Povezave do sosedov vozlišča t imajo pristranskost $\alpha_{pq}(t, x) = 1$. Ostale povezave imajo pristranskost $\alpha_{pq}(t, x) = 1/q$.

$p > \max(q, 1)$, potem bo naključni sprehod malo verjetno generiral vozlišča, ki smo jih že obiskali. V tem primeru bo okolica vsebovala vozlišča, ki so bolj oddaljena od u, kot če bi veljalo $p < \max(q, 1)$.

Spreminjanje oblike okolice je koristno, ker vpliva na končne rezultate. Omrežja lahko vsebujejo informacije, ki jih je težko razbrati le z enim tipom okolice, prav tako le en tip okolice ni primeren za vsa omrežja. S spreminjanjem parametrov p in q lahko metodo node2vec prilagodimo različnim omrežjem in različnim tipom informacij, ki jih želimo dobiti.

3.2.5 Izbira značilk

Po luščenju značilk za vsak članek dobimo veliko število značilk, kar je lahko težava, saj pri velikem številu značilk nekateri algoritmi strojnega učenja ne delujejo najbolje. Za dobre rezultate je potrebno izbrati majhen del značilk, ki dajo najboljše rezultate. Uporabimo lahko različne statične metode, ki poiščejo podmnožico značilk, ki najbolje opisujejo problem.

Izbiro značilk smo opravili z metodo ANOVA (analysis of variance) [17]. Metoda ANOVA nam za posamezno značilko pove, ali so razlike med povprečji vrednosti značilke pri različnih razredih dovolj različne. Pri metodi predpostavimo, da so podatki razdeljeni v razrede in ustrezajo statističnemu modelu $Y_{ij} = \mu_i + \varepsilon_{ij}$, kjer je Y_{ij} j-ti primer i-tega razreda, μ_i aritmetična sredina razreda i in ε_{ij} naključno odstopanje Y_{ij} od μ_i. Predpostavimo, da so ε_{ij} neodvisne naključne spremenljivke z Gaussovo porazdelitvijo $\mathcal{N}(0, \sigma^2)$. Za posamezen razred izračunamo varianco po enačbi

$$s_i^2 = \frac{1}{n-1} \sum_{j=0}^{n} (Y_{ij} - \mu_i)^2$$

S statističnim F testom ocenimo, ali ima značilka pri dveh različnih razredih povprečji, ki sta si statistično dovolj različni:

$$F = \frac{s_1^2}{s_2^2}$$

Na podlagi tega zavržemo značilke, ki nimajo dovolj različnih povprečij med različnimi razredi.

Preizkusili smo metodo vzajemne informacije (mutual information) [31], s katero lahko izračunamo mero odvisnosti med različnimi značilkami:

$$I(X,Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) log(\frac{p(x,y)}{p(x)p(y)}),$$

kjer je $I(X,Y)$ vzajemna informacija spremenljivk X in Y. Manjši je faktor $I(X,Y)$, bolj sta spremenljivki X in Y neodvisni. Na podlagi tega izberemo značilke, ki so čimbolj neodvisne. Medsebojno odvisnih značilk ne želimo, saj ne dajo dodatnih informacij. Ker večje število spremenljivk zmanjša uspešnost algoritmov strojnega učenja takšne značilke odstranimo.

Metoda ANOVA ne upošteva odvisnosti med značilkami. Lahko se zgodi, da dve značilki samostojno slabo ločujeta med razredi, obe skupaj pa ločujeta dobro. Odvisnosti med spremenljivkami lahko zaznamo z metodo Relief [29], ki izbere najkoristnejše spremenljivke in pri tem upošteva odvisnosti med

njimi. Za posamezen primer R_i definira najbližji zadetek H, kot najbližji primer z istim razredom, in najbližjo zgrešitev M, kot najbližji primer z različnim razredom. Pomembnosti značilk posodobi glede na njihove vrednosti v R_i, H in M. Postopek ponovimo za m naključno izbranih primerih R_i. Psevdokoda algoritma je predstavljena v algoritmu 1.

Algorithm 1 Psevdokoda algoritma Relief

```
W[A] = 0 za vsak A
for i = 1 to m do
    R_i ← naključen primer
    H ← najbližji primer R_i istega razreda
    H ← najbližji primer R_i različnega razreda
    for A = 1 to a do
        W[A] := W[A] − diff(A, R_i, H)/m + diff(A, R_i, M)/m
    end for
end for
```

Funkcija diff(A, I_1, I_2), ki posodobi pomembnosti spremenljivk $W[A]$, je definirana kot:

$$\text{diff}(A, I_1, I_2) = \frac{|value(A, I_1) - value(A, I_2)|}{max(A) - min(A)}$$

Metoda Relief je omejena na dvorazredno klasifikacijo in ne definira postopka za obravnavanje manjkajočih podatkov. V našem delu smo uporabili nadgradnjo ReliefF [30], ki te pomanjkljivosti odpravi. Namesto enega zadetka in ene zgrešitve poišče k zadetkov in k zgrešitev, kjer so zgrešitve primeri iz razredov, različnih od R_i. Uporabi tudi različno funkcijo diff, ki deluje z manjkajočimi podatki.

3.3 Gručenje z omejitvami

Po ekstrakciji in izboru značilk je vsak članek opisan z naborom značilk, ki opisujejo njegovo vsebino. Na podlagi teh značilk želimo članke združiti v gruče tako, da se vsebinsko podobni članki nahajajo v istih gručah. Želimo,

da dobljene gruče ustrezajo strukturi urnika, ki je definirana pred začetkom gručenja. Struktura urnika definira dve omejitvi: število gruč in velikost posamezne gruče.

Standardni algoritmi gručenja, kot sta metoda k-voditeljev (k-means clustering) [23] in hiearhično gručenje [26], ne upoštevajo obeh omejitev. Pri obeh lahko določimo število gruč, ne pa tudi njihovih velikosti. Algoritmi gručenja z omejitvami, kot so COP k-means [56], omejitve definirajo med posameznimi primeri, in ne na nivoju gruč. Pogosto omejitve navedejo, kateri primeri se morajo nahajati v isti gruči in kateri se ne smejo. Če želimo definirati velikosti posameznih gruč, potrebujemo drugačen pristop.

Nekateri algoritmi gručenja z omejitvami namesto omejitev na nivoju primerov definirajo omejitve uravnoteženosti [58]. Takšni algoritmi vrnejo gruče, ki so enakih velikosti. Podoben pristop lahko uporabimo, če želimo gruče z različnimi, točno definiranimi velikostmi.

Za gručenje člankov v gruče z znanimi velikostmi smo uporabili modificirano metodo k-voditeljev, ki je podobna metodi opisani v [20]. Metoda poteka v dveh korakih:

1. Inicializacija: Pred začetkom algoritma vsaki gruči $c_1, c_2, ..., c_n$ priredimo maksimalno velikost $S_1, S_2, ..., S_n$. Kot pri metodi k-voditeljev vsaki gruči priredimo začetnega voditelja, ki predstavlja centroid gruče. Za izbiro začetnih voditeljev obstajajo različni pristopi, mi uporabimo metodo k-means++ [8], predstavljeno v algoritmu 2. Začetne voditelje lahko izberemo tudi povsem naključno iz primerov podatkovne množice, vendar metoda k-means++ izboljša rezultate gručenja.

Po inicializaciji začetnih voditeljev razvrstimo vse primere x_i iz podatkovne množice glede na razliko med razdaljo do najbližjega in najbolj oddaljenega voditelja d_i, ki nam pove, kateri primeri bodo povzročili največjo napako v primeru najslabše razvrstitve: $d_i = \min_j \|x_i - c_j\| - \max_j \|x_i - c_j\|$

Primeri z nizkim d_i bodo povzročili veliko napako, če jih dodelimo napačni gruči. Primere z d_i blizu 0 lahko razvrstimo v poljubno gručo brez velike na-

Algorithm 2 Psevdokoda algoritma k-means++

```
1: c_0 naj bo naključen primer iz podatkovne množice X
2: j = 1
3: repeat
4:     for all x ∈ X do
5:         D(x) ← min_{i∈[0,j)} ||x − c_i||
6:     end for
7:     j ← j + 1
8:     c_j naj bo naključen primer iz X, izbran z verjetnostjo proporcionalno D(x)^2
9: until j = želeno število gruč
```

pake. V zadnjem koraku inicializacije primere, glede na njihovo razvrstitev po vrednosti d_i, enega po enega poskusimo dodeliti njihovim najbližjim centroidom. Če gruča centroida, ki mu želimo dodeliti primer, še ni polna, mu primer dodelimo. Če je gruča polna centroid označimo za polnega in ponovno izračunamo vrednosti d_i za še ne dodeljene primere primere, pri čimer ne upoštevano polnih gruč. Nedodeljene primere ponovno razvrstimo po vrednosti d_i in ter nadaljujemo s postopkom dodeljevanja.

2. Optimizacija: V drugem koraku izhajamo iz že obstoječe razvrstitve primerov v gruče. Izvedemo lokalno optimizacijo, s katero želimo izboljšati začetno stanje. Kot mero napake definiramo vsoto razdalj primerov x_i do centroidov njihovih gruč $c[i]$:

$$E = \sum \|x - c[i]\|$$

Optimizacija poteka v sledečih korakih:

i.) Izračun centroidov - Voditelje gruč c_i prestavimo tako, da se nahajajo na sredini njihove gruče. Veljati mora:

$$c_i = \frac{1}{|c_i|} \sum_{x_j \in c_i} x_j$$

ii.) Izračun razdalj do centroida - Za vsak primer x izračunamo njegovo

razdaljo do vseh centroidov:

$$D(x, c) = \|x - c\|$$

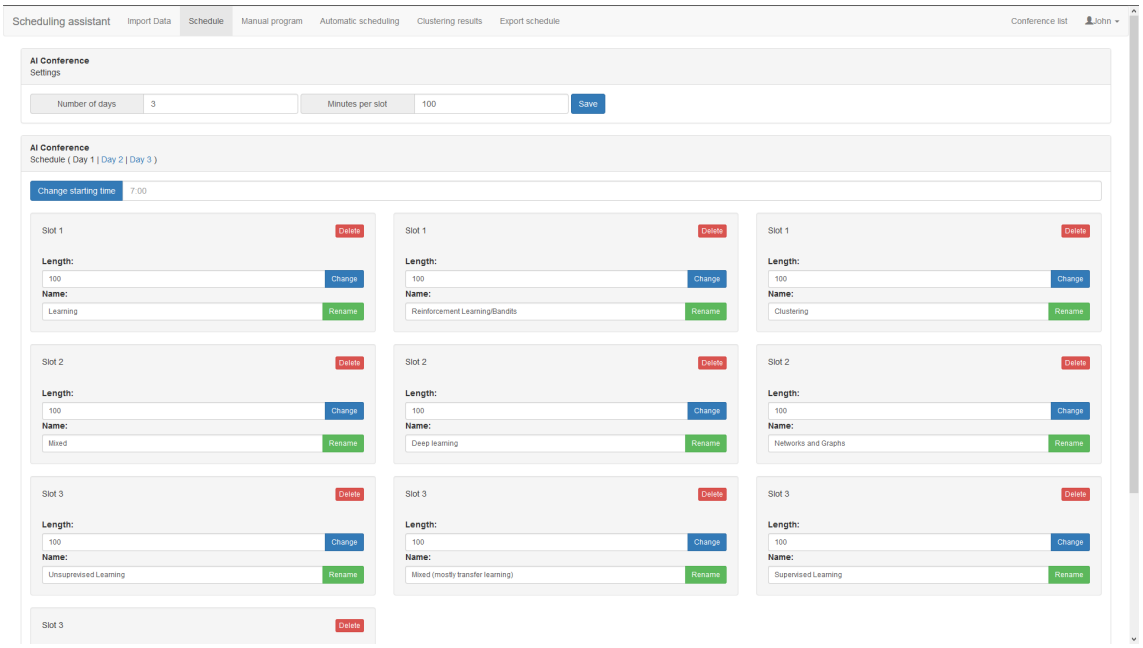
iii.) Razvrstitev primerov - Primere razvrstimo glede na razliko razdalje do centroida njegove trenutne gruče $D(x, c)$ in najbližjega centroida izven njegove gruče $D'(x, c)$. Primeri, kjer je razdalja $D(x, c) - D'(x, c)$ največja so, trenutno najslabše razvrščeni.

iv.) Zamenjava primerov - Zamenjamo gruče tistim parom primerov, kjer taka zamenjava zmanjša razdalji do centroidov gruč obeh primerov. Pri tem najprej zamenjamo primere z veliko vrednostjo $D(x, c) - D'(x, c)$. Psevdokoda za zamenjavo je predstavljena v algoritmu 3.

Algorithm 3 Psevdokoda zamenjave primerov pri gručenju z omejitvami.

```
1: L ← [] //L je množica primerov, ki želijo zamenjati gručo
2: X ← množica primerov, urejena glede na D(x,c) − D'(x,c)
3: for all x ∈ X do
4:    c ← centroid gruče, ki ji pripada x
5:    if L ≠ [] then
6:       for all x' ∈ L do
7:          c' ← centroid gruče, ki ji pripada x'
8:          if (D(x,c') < D(x,c))&(D(x',c) < D(x',c')) then
9:             x priredimo gruči c'
10:            x' priredimo gruči c
11:            break
12:         end if
13:      end for
14:   end if
15:   če x še vedno pripada gruči c, ga dodamo v L
16: end for
```

Korake optimizacije ponavljamo, dokler ni več možnih zamenjav. Ker zamenjavo izvedemo le, če se obema primeroma zmanjša razdalja do centroida gruče se bo metoda zagotovo ustavila. Metoda se ustavi v lokalnem minimumu napake E, ne zagotavlja pa globalnega minimuma, zaradi česar lahko



Slika 3.5: Spletni vmesnik aplikacije, v okviru katere je implementirana metoda za samodejno razvrščanje člankov v urnik konference.

rezultate izboljšamo tako, da jo večkrat ponovimo z različnimi začetnimi centroidi.

3.4 Spletna aplikacija

Algoritem za samodejno razvrščanje člankov v urnik konference smo implementirali v spletni aplikaciji, razviti v diplomskem delu [51]. Spletna aplikacija organizatorjem konferenc omogoča izgradnjo strukture urnika in uporabo postopka samodejnega razvrščanja člankov preko spletnega vmesnika. Aplikacija je bila zgrajena z uporabo spletnega ogrodja Django [1]. Vmesnik spletne aplikacije je prikazan na sliki 3.5.

Poglavje 4

Evalvacija in rezultati

Kakovost samodejnega razporejanja člankov v urnik je težko oceniti. Ker je kakovost urnikov konferenc subjektivna, ne obstajajo uveljavljene mere, s katerimi bi lahko ocenili kakovost urnika. Tudi postopek iskanja podobnih člankov je težko oceniti. Za ocenjevanje uspešnosti podobnih algoritmov se običajno uporabijo podatkovne baze, kjer so ročno označeni atributi primerov, ki jih želimo napovedati. Za znanstvene članke bi torej potrebovali podatkovno bazo člankov z ročno označenim znanstvenim podpodročjem članka, vendar takšna prosto dostopna podatkovna baza ne obstaja. Članke je težko označiti z njihovim znanstvenim podpodročjem, saj veliko člankov spada na več podpodročij.

Evalvacijo smo izvedli s kombinacijo različnih pristopov. Za ocenjevanje kakovosti iskanja podobnih člankov smo zgradili ročno označeno podatkovno bazo člankov s področja strojnega učenja. Zgrajeno bazo smo uporabili tudi za oceno učinkovitosti posameznih metod luščenja značilk. Za posamezne metode luščenja značilk smo izvedli samostojno evalvacijo z metodami specifičnimi za luščeno značilko. Kakovost končnega urnika smo ocenili glede na to, koliko člankov v isti seji si je delilo znanstveno podpodročje. Poleg tega smo samodejno razvrščanje člankov v urnik preizkusili na dejanski izvedbi konference ECML-PKKD 2017.

V poglavju najprej podrobneje predstavimo različne metode evalvacije

uporabljenih metod. V razdelku 4.1 predstavimo ročno označeno podatkovno bazo, ki smo jo zgradili za potrebe testiranja uporabljenih metod. V razdelku 4.2 predstavimo, kako smo ocenili kakovost značilk, uporabljenih za iskanje podobnih člankov. V razdelku 4.4 predstavimo evalvacijo algoritma, ki na podlagi izbranih značilk članke razporedi v vnaprej definirane seje urnika tako, da se podobni članki nahajajo skupaj. V razdelku 4.3 predstavimo evalvacijo metod za analizo grafov. V razdelku 4.5 predstavimo testiranje na konferenci ECML-PKKD 2017.

4.1 Podatkovna baza

Zgradili smo ročno označeno podatkovno bazo akademskih člankov. Vsak primer v bazi je sestavljen iz besedila članka in področja, na katerega spada. Pri izgradnji podatkovne baze smo se omejili na članke s področja strojnega učenja. Uporabili smo članke, predstavljene na konferencah ECML-PKKD 2015, ECML-PKKD 2016 in ICML 2016. Uporabili smo tudi 200 člankov iz drugih konferenc s področja strojnega učenja (SIGKDD, AISTATS in NIPS). Število člankov iz vsake konference je predstavljeno v tabeli 4.1. Vsak članek smo označili s tematskim področjem, na katerega spada. Članki iz konferenc ECML-PKKD 2015, ECML-PKKD 2016 in ICML 2016 so bili označeni s področji, pod katerimi so bili navedeni v urniku konference. Članki iz drugih konferenc so bili ročno umeščeni v ustrezna področja. Za lažje ročno označevanje člankov smo se pri člankih iz drugih konferenc omejili na članke z jasno razvidnimi področji. Področja člankov, predstavljenih na konferencah so prikazana v tabeli 4.2.

Ker področja na konferencah ECML-PKKD 2015, ECML-PKKD 2016 in ICML 2016 niso povsem enaka, smo kot razred uporabili področja s konference ICML 2016, saj je vsebovala največje število razredov. Področja so si dovolj podobna, da smo lahko članke z ostalih konferenc ustrezno označili s področji s konference ICML 2016.

Tabela 4.1: Število člankov v podatkovni bazi glede na konferenco.

Konferenca	Število člankov v podatkovni bazi
ECML-PKKD 2015	63
ECML-PKKD 2016	90
ICML 2016	262
Ostale konference	200
Skupaj	615

4.2 Evalvacija luščenja značilk

Značilke smo ocenili na dva ločena načina:

- z uporabo statističnih metod, ki nam povejo, kako razpršene so vrednosti značilk glede na različne razrede;
- z učenjem modelov strojnega učenja na različnih naborih značilk in primerjavo rezultatov klasifikacije s temi modeli.

Pri statističnih metodah smo uporabili enak postopek kot pri izboru značilk. Kakovost značilk smo ocenili z metodama ANOVA in in ReliefF ter z vzajemno informacijo. Metoda ReliefF je delovala počasi in ni vrnila boljših rezultatov od metode ANOVA, kar je veljalo tudi za vzajemno informacijo. Tabela 4.3 prikazuje, koliko značilk posameznega tipa je metoda ANOVA izbrala za najboljših 1000 značilk. Matrika doc2vec in metoda personalized PageRank nista delovala dobro, graf referenc in vektorji terminologije pa so vsebovali koristne podatke. Poleg tega smo zgradili klasifikacijski model iz vsakega tipa značilk posebej in značilke ocenili glede na klasifikacijsko točnost modela.

Modele strojnega učenja smo zgradili na sledečih naborih značilk:

- samo matrika tf-idf,
- samo graf referenc, pretvorjen v vektorsko obliko z metodo node2vec,
- vektorji izluščene terminologije,

Tabela 4.2: Področja konferenc ECML-PKKD 2015 in 2016 ter ICML 2016.

Področja ECML-PKKD 2015	Področja ECML-PKKD 2016	Področja ICML 2016
Rich data	Deep learning	Approximate Inference
Probabilistic Models	Neural Networks	Bandit Learning
Probabilistic Methods	Graphs	Bayesian nonparametrics
Statistical Methods	Clustering	Bayesian optimization
Data Streams	Learning	Causality
Online Learning	Classification	Clustering
Sparsity	Optimization	Computational Advertising
Compressed Sensing	Topic Modelling	Social Science
Matrix Analysis	Patterns	Deep Learning
Tensor Analysis	Kernels	Deep Learning and Vision
Model Evaluation	Probabilistic learning	Deep Learning Computations
Selection	Data science	Distributed Optimization
Classification	Social Networks	Feature Selection
Supervised Learning	Reinforcement learning	Gaussian Processes
Graph Learning	Factorization	Hashing
Graphical Models	Streams	Kernel Methods
Bayesian Networks	Time Series	Large Scale Learning
Data Preprocessing	Semi suprevised learning	Learning Theory
Pattern Mining	Dimensionality	Manifold learning
Sequence Mining	Patterns in sequences	Matrix Factorization
Social Learning	Bandits	Monte Carlo Methods
Graph Learning	Transfer Learning	Natural Language Processing
Clustering	Rich data	Networks and Graphs
Unsupervised Learning		Online Learning
Deep Learning		Optimization
Regression		Privacy
Bandits		Probabilistic methods
Large Scale Learning		Ranking learning
Big Data		Reinforcement Learning
Preference Learning		Sparse optimization
Multi-task Learning		Sparsity
Transfer Learning		Structured prediction
Metalearning		Submodularity
Distance Learning		Supervised Learning
Metric Learning		Time Series Analysis
		Topic Models
		Transfer Learning
		Unsupervised Learning
		Variational Inference
		Vision

Tabela 4.3: Razporeditev najboljših značilk glede na tip značilke.

Tip značilke	Število značilk v najboljših 200	Število vseh značilk tipa
Matrkia tf-idf	965	32762
Graf referenc (node2vec)	29	128
Graf referenc (P-PR)	0	128
Vektorji terminologije	6	100
Matrika doc2vec	0	128

- matrika doc2vec,
- graf recenzentov, pretvorjen v vektorsko obliko z metodo node2vec,
- vse značilke,
- različni nabori najboljših x spremenljivk, izbranih z metodo ANOVA.

Za vsak nabor značilk smo zgradili klasifikacijske modele z uporabo logistične regresije, naključnih dreves in metode podpornih vektorjev, ki so tri pogosto uporabljene metode strojnega učenja. Modele smo testirali z uporabo 10-kratnega prečnega preverjanja. S tem smo dobili rezultate, predstavljene v tabeli 4.4. Graf recenzentov smo testirali na ločeni podatkovni bazi, saj smo imeli podatke o mnenju recenzentov na voljo le za članke predstavljene na konferenci ECML-PKKD 2017. Iz tabele je razvidno, da med posameznimi skupinami značilk najbolje deluje graf referenc in izluščena terminologija. Metoda doc2vec doseže rezultate, ki niso boljši od matrike tf-idf. Slabo deluje tudi graf recenzentov, vendar je slab rezultat verjetno posledica testiranja na manjši podatkovni bazi. Z zlivanjem različnih tipov značilk smo rezultat izboljšali. Najboljše rezultate smo dosegli z metodo naključnih dreves z 1000 najboljšimi značilkami. Večinski klasifikator ima klasifikacijsko točnost 0.061.

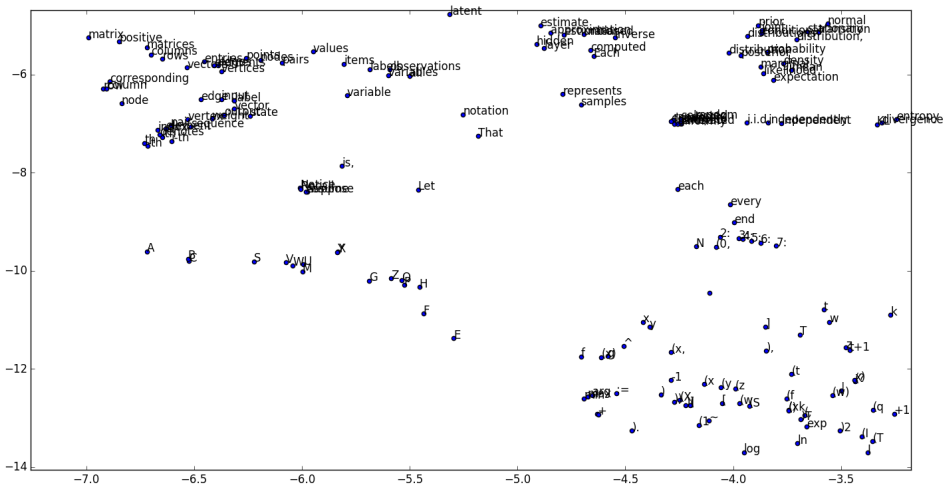
Poleg metode doc2vec smo ločeno testirali delovanje metode word2vec na naši podatkovni množici. Ker metoda word2vec ne vrne vektorjev, ki bi bili uporabni za klasifikacijo, je nismo mogli testirati z zgornjimi metodami.

Tabela 4.4: Rezultati različnih klasifikacijskih modelov.

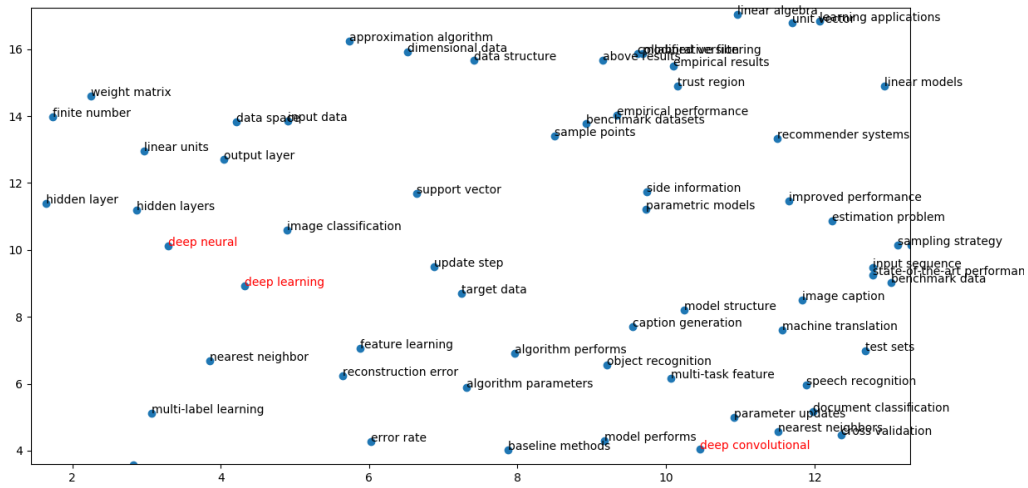
		Povprečne klasifikacijske točnosti	
Uporabljene značilke	Logistična regresija	RF	SVM
tf-idf	0.287	0.402	0.34167
Graf referenc	0.413	0.434	0.34179
Terminologija	0.407	0.319	0.34147
Doc2vec	0.284	0.292	0.34154
Graf recenzentov	0.285	0.290	0.340
Vse značilke	0.401	0.414	0.34154
Najboljših 200	0.284	0.425	0.34192
Najboljših 1000	0.457	**0.523**	0.34147
Najboljših 2000	0.480	0.515	0.34154
Najboljših 5000	0.481	0.492	0.34199
Najboljših 10000	0.481	0.497	0.34154

Namesto tega smo dobljene besedne vektorje vizualizirali in preverili, ali so dobljeni rezultati smiselni. Rezultati vizualizacije so prikazani na sliki 4.1. Na sliki smo dimenzijo vektorskih vložitev zmanjšali iz 300 na 2 z uporabo metode t-SNE [35]. Iz slike je razvidno, da metoda ustrezno loči nekatere tipe besed. Besede, ki se pojavijo v enačbah (*log*, *ln*, *exp*) razvrsti v svojo gručo. Prav tako v svojo gručo razvrsti posamezne črke.

Po odstranitvi manj koristnih gruč, kot so gruča ločil, črk in besed, ki se pojavijo v enačbah, metoda word2vec ostale besede dobro razvrsti glede na njihov pomen. Slika 4.2 prikazuje metodo word2vec uporabljeno na bigramskih terminoloških izrazih, ki smo jih dobili z luščenjem terminologije. Slika prikazuje okolico terminoloških izrazov, ki vključujejo besedo *deep*. V okolici teh izrazov se nahajajo tudi drugi izrazi, ki so povezani s področjem globokega učenja (na primer *image classification*, *hidden layer*, *output layer*, *linear units* in *update step*).



Slika 4.1: Rezultati vizualizacije metode word2vec.



Slika 4.2: Vizualizacija metode word2vec na terminoloških izrazih pridobljenih z luščenjem terminologije. Slika prikazuje okolico terminoloških izrazov, ki vključujejo besedo *deep*.

4.3 Evalvacija analize omrežij

Pri evalvaciji metod analize omrežij smo ocenili, kako različni parametri uporabljeni pri izgradnji omrežij vplivajo na končni rezultat. Uporabljene metode izgradnje grafov vsebujejo sledeče parametre:

Omrežje bibliografskega sklapljanja - V omrežje bibliografskega sklapljanja sta članka povezana, če si delita vsaj n referenc. S spreminjanjem parametra n lahko vplivamo na število povezav, s čimer lahko odstranimo možne moteče povezave.

Omrežje terminologije - Pri omrežju terminologije lahko na velikost in obliko grafa vplivamo na dva načina. Iz vsakega članka izluščimo n najverjetnejših terminoloških izrazov. Dva terminološka izraza sta povezana, če se pojavita v istem članku vsaj m-krat. Poleg tega lahko na obliko grafa vplivamo s spreminjanjem parametrov pri luščenju terminologije.

Omrežje mnenje recenzentov - Pri omrežju mnenja recenzentov lahko določimo uteži med povezavami člankov. Ker smo imeli pri grafu recenzentov za testiranje na voljo le majhno testno množico vpliva različnih uteži na končne rezultate nismo preverili.

Vpliv parametrov na končni rezultat smo preverili na dva načina:

- Omrežja smo primerjali glede na njihovo strukturo. Pri dobrih omrežjih bibliografskega sklapljanja se podobni članki nahajajo blizu drug drugega. V dobrih omrežjih terminologije se terminološki izrazi z istega članka nahajajo blizu drug drugega.

- Omrežja zgrajena z različnimi parametri smo primerjali glede na klasifikacijsko točnost modela, ki je za klasifikacijo uporabljal značilke pridobljene iz omrežij.

Pri primerjavi struktur omrežij bibliografskega sklapljanja smo primerjali povprečno razdaljo med dvema člankoma z istega področja in razdaljo med

Tabela 4.5: Rezultati testiranja parametrov pri izgradnji omrežja bibliografskega sklapljanja.

n	Razdalja med člankoma istega področja	Razdalja med naključnima člankoma
1	avg = 1.556, std = 0.641	avg = 2.136, std = 0.381
2	avg = 1.556, std = 0.641	avg = 2.077, std = 0.384
3	avg = 1.461, std = 0.667	avg = 1.995, std = 0.499
4	avg = 0.508, std = 0.383	avg = 1.152, std = 0.477
5	avg = 0.271, std = 0.130	avg = 0.358, std = 0.136
6	avg = 0.185, std = 0.089	avg = 0.277, std = 0.106

dvema naključnima člankoma. Ker se s spreminjanjem parametra n spreminja velikost omrežja, razdalja med članki z istega področja ni zadosten kriterij. Rezultati primerjave so prikazani v tabeli 4.5. Največje razmerje med razdaljami znotraj področja in med naključnimi članki dobimo pri vrednosti $n = 4$. Za $n > 6$ dobimo graf z majhnim številom povezav, ki vrne slabe rezultate.

Pri primerjavi omrežij terminologije smo primerjali povprečno razdaljo med n najverjetnejšimi terminološkimi izrazi iz člankov z istega znanstvenega področja in n najverjetnejšimi izrazi iz člankov z naključnih področij. Poleg parametra n smo primerjali rezultate glede na parameter m, ki vpliva na strukturo omrežja. V omrežju sta dva terminološka izraza povezana, če se v istem članku pojavita vsaj m-krat. Rezultati so prikazani v tabeli 4.6. Iz tabele je razvidno, da je omrežje terminologije občutljivo na spremembe parametrov. Za dobre rezultate je potrebno dobro izbrati parametre. Iz tabele je razvidno, da dobimo boljše rezultate če se omejimo na majhno število najverjetnejših rezultatov. Razlike med razdaljami so majhne, vendar so z dobro nastavljenimi parametri v povprečju nižje pri terminoloških izrazih člankov z istega področja.

Tabela 4.6: Rezultati testiranja parametrov pri izgradnji omrežja terminologije.

m	n	Razdalja med termini istega članka	Razdalja med termini naključnih člankov
1	10	avg = 3.0, std = 2.4	avg = 4.4, std = 2.2
3	10	avg = 3.861, std = 0.77	avg = 4.026, std = 0.82
3	50	avg = 3.922, std = 0.41	avg = 3.859, std = 0.37
7	10	avg = 2.50, std = 1.550	avg = 3.14, std = 1.772
7	50	avg = 2.470, std = 0.641	avg = 2.536, std = 0.381
14	10	avg = 7.920, std = 1.27	avg = 8.423, std = 0.59
14	50	avg = 8.211, std = 1.34	avg = 8.776, std = 0.73

4.4 Evalvacija gručenja z omejitvami

Pri gručenju z omejitvami želimo podobne članke razvrstiti v skupine tako, da se članki z istega raziskovalnega področja nahajajo v istih skupinah. Skupine, ki jih vrne gručenje, se morajo ujemati z urnikom konference. To pomeni, da imamo podano omejitev glede števila skupin in velikosti vsake skupine.

Uspešnost takšnega gručenja lahko ocenimo tako, da preverimo kako podobni so si članki znotraj vsake skupine. To smo storili z uporabo metode silhoutte [48]. Za vsak primer iz podatkovne baze i definiramo $a(i)$ kot povprečno razdaljo med i in ostalimi primeri iz iste gruče in $b(i)$ kot razdaljo med i in gručo, ki je najbližje primeru i in ga ne vsebuje. Razdaljo med primerom i in gručo definiramo kot povprečno razdaljo med primerom i in vsemi točkami v gruči. Za vsak primer nato definiramo vrednost silhouette:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Velja $-1 \leq s(i) \leq 1$. Če je vrednost $s(i)$ blizu 1, potem je bil primer razvrščen v dobro gručo. Če je $s(i)$ blizu -1 bi bilo bolje, če bi bil primer razvrščeno v drugo gručo. Oceno celotnega gručenja dobimo tako, da vzamemo povprečje $s(i)$ čez vse primere. Kakovost posameznih gruč smo ocenili tudi tako, da smo vzeli povprečje $s(i)$ primerov v tej gruči.

Tabela 4.7: Vrednosti silhouette različnih gručenj.

Način gručenja	Vrednost silhouette
k-voditeljev s 5 gručami	0.217
k-voditeljev s 35 gručami	0.080
k-voditeljev z omejitvami velikosti gruč in 35 gručami	-0.212
Naključno gručenje	-0.316

Samostojna meritev silhouette ni dobra ocena za gručenje z omejitvami velikosti gruč, saj metoda silhouette ne upošteva teh omejitev. Zaradi tega smo primerjali vrednosti silhouette med več različnimi gručenji. Rezultati so predstavljeni v tabeli 4.7. Najboljše rezultate dobimo z metodo k-voditeljev in majhnim številom gruč, ki vse razen štirih primerov razvrsti v eno gručo. Takšno gručenje kljub najvišji vrednosti silhouette ni dobro. Metoda k-voditeljev s 35 gručami razvrsti večino kandidatov v majhno število gruč, okoli polovica gruč pa vsebuje le en primer. Gručenje z omejitvami velikosti gruč zniža vrednost silhouette, vendar zagotovi, da so gruče velikosti, ki ustrezajo sejam urnika. Naključno gručenje vrne najslabše rezultate.

4.5 Evalvacija na člankih s konference

Celoten postopek samodejnega razvrščanja člankov v urnik smo v sodelovanju z organizatorji evalvirali na člankih sprejetih na konferenco ECML-PKKD 2017. Za evalvacijo smo imeli na voljo strukturo urnika, vse članke, ki bodo predstavljeni na konferenci, in mnenja recenzentov izražena v postopku izbire člankov. Na podlagi tega smo razvrstili članke v urnik konference. Na konferenci bo predstavljenih 136 člankov s področja strojnega učenja. Struktura urnika je predstavljena v tabeli 4.8. Vse seje trajajo 100 minut, razen štirih v drugem dnevu, ki trajajo 60 minut.

S samodejnim razvrščanjem člankov smo pridobili urnik konference. Poleg razvrstitve člankov v urnik smo vsakemu članku napovedali tudi področje, na katerega spada. Tabela 4.9 prikazuje področja sej zgrajenega urnika. Če

Tabela 4.8: Struktura urnika konference ECML-PKKD 2017.

Dan 1			
Seja 1	Seja 2	Seja 3	
Seja 4	Seja 5	Seja 6	
Seja 7	Seja 8	Seja 9	Seja 10
Dan 2			
Seja 11	Seja 12	Seja 13	
Seja 14	Seja 15		
Seja 16 (60 min)	Seja 17 (60 min)	Seja 18 (60 min)	Seja 19 (60 min)
Dan 3			
Seja 20	Seja 21	Seja 22	
Seja 23	Seja 24	Seja 25	
Seja 26	Seja 27	Seja 28	Seja 29

seja vsebuje večino člankov, ki jih je klasifikator označil z enakim razredom, potem je seja označena z imenom tega razreda. Drugače je označena z besedo *mešano*. V celotnem urniku je 12 mešanih sej in 17 sej, kjer večina člankov spada na isto podpodročje. Mešane seje vsebujejo članke z bolj specifičnimi razredi, kot so na primer *Privacy*, *Kernel methods* in *Natural language processing*. Članki s teh podpodročij velikokrat vsebujejo pristope iz bolj splošnih podpodročij, kot sta globoko učenje in gručenje, zaradi česar je težko zaznati njihovo pravo podpodročje. Zgrajen urnik ni primeren za uporabo na konferenci, saj vsebuje preveč mešanih sej, lahko pa služi kot izhodišče za izgradnjo končnega urnika.

Tabela 4.9: Seje urnika pridobljenega s samodejnim razvrščanjem člankov v urnik.

Dan 1			
Learning	Reinforcement Learning/Bandit Learning	Clustering	
Mešano	Deep learning	Networks and graphs	
Unsupervised learning	Mešano	Supervised learning	Gaussian processes
Dan 2			
Mešano	Networks and graphs	Supervised learning	
Mešano	Mešano		
Bandit learning	Supervised learning	Mešano	Mešano
Dan 3			
Transfer learning	Mešano	Bayesian methods	
Mešano	Clustering	Mešano	
Supervised learning	Mešano	learning theory	Mešano

Poglavje 5

Zaključek

V magistrski nalogi smo razvili metodo za samodejno razvrščanje člankov v urnik konference. Z uporabo metod iz različnih področij strojnega učenja smo razvili postopek za iskanje pomensko podobnih člankov. Poleg besedil člankov smo si pri iskanju podobnosti pomagali z informacijami v obliki grafov. Te podatke smo najprej pretvorili v vektorsko obliko. Za to smo testirali dve različni metodi – Personalized PageRank in node2vec. Na podlagi podobnosti smo članke razvrstili v seje urnika konference tako, da se podobni članki nahajajo v istih sejah. Za razvrščanje člankov v seje smo uporabili prilagojen algoritem k-voditeljev ki upošteva omejitve velikosti gruč. Samodejno razvrstitev člankov v urnik konference smo implementirali kot del spletne aplikacije, ki organizatorjem konference omogoča preprosto uporabo postopka.

Razvit postopek smo testirali na več načinov. Posamezne uporabljene metode smo testirali z različnimi splošno uveljavljenimi testi. Za evalvacijo celotnega postopka smo ustvarili ročno označeno podatkovno bazo znanstvenih člankov. Postopek smo v celoti testirali tudi na člankih konference ECML-PKKD 2017. Testirali smo učinkovitost posameznih značilk, ki smo jih uporabili za iskanje podobnih člankov. V rezultatih smo pokazali, da za iskanje podobnih člankov dobro delujejo metode ekstrakcije terminologije in analize grafov.

Razvit postopek bi lahko izboljšali na več načinov. Za iskanje podobnih člankov bi lahko testirali dodatne metode luščenja značilk z uporabo metod semantične analize in analize omrežij. Uporabili bi lahko tudi različne tipe značilk, kot so ključne besede, ki jih avtorji vnesejo pri oddaji člankov. Lahko bi izvedli dodatno predobdelavo besedil in odstranili pomensko nekoristne dele člankov, kot na primer enačbe. Za gručenje z omejitvami smo uporabili modificiran algoritem k-voditeljev, ki popolnoma zadosti omejitvam glede velikosti gruč, vendar pri optimizaciji gručenja zagotavlja le lokalni minimum. Pri gručenju z omejitvami bi lahko uporabili metode, kot sta CSCLP (Clustering Algorithm with Size Constraints and Linear Programming) [54] in K-MedoidsSC (K-Medoids algorithm with Size Constraints) [54].

Izboljšali bi lahko tudi evalvacijo postopka. Postopek smo v celoti testirali na ročno zgrajeni podatkovni množici, ki je vsebovala članke s področja strojnega učenja. Postopka nismo testirali na drugih področjih.

Pretvorbo metapodatkov v obliki grafov v vektorsko obliko in nekatere uporabljene metode za iskanje podobnih člankov so bile objavljene v sledečem članku:

Tadej Škvorc, Nada Lavrač, and Marko Robnik-Šikonja. Co-bidding graphs for constrained paper clustering. In *5th Symposium on Languages, Applications and Technologies*, 2016

Literatura

[1] The web framework for perfectionists with deadlines — django. Dostopno na `https://www.djangoproject.com/`, 2015. (dostopano 7. 9. 2017).

[2] Easychair home page. Dostopno na `http://easychair.org/`, 2017. (dostopano 7. 9. 2017).

[3] Edas: Editor's assistant. Dostopno na `https://edas.info/doc/`, 2017. (dostopano 7. 9. 2017).

[4] Iapr commence conference management system. Dostopno na `http://iaprcommence.sourceforge.net/`, 2017. (dostopano 7. 9. 2017).

[5] Microsoft's conference management toolkit. Dostopno na `https://cmt.research.microsoft.com/cmt/`, 2017. (dostopano 7. 9. 2017).

[6] Openconf peer-review, conference and abstract management software system. Dostopno na `http://www.openconf.com/`, 2017. (dostopano 7. 9. 2017).

[7] Xpdf: Home. Dostopno na `http://www.foolabs.com/xpdf/home.html`, 2017. (dostopano 7. 9. 2017).

[8] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[9] Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

[10] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[11] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

[12] Mark Davies. The corpus of contemporary american english (COCA): 520 million words, 1990-present, 2008.

[13] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

[14] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Knowledge discovery in databases*, volume 96, pages 226–231, 1996.

[15] Christiane Fellbaum. *WordNet*. Wiley Online Library, 1998.

[16] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 2016.

[17] David A Freedman. *Statistical models: theory and practice*. Cambridge University Press, 2009.

[18] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

[19] Keinosuke Fukunaga and Larry D Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1975.

[20] Nuwan Ganganath, Chi-Tsun Cheng, and K Tse Chi. Data clustering with cluster size constraints using a modified k-means algorithm. In *2014 IEEE International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 158–161. IEEE, 2014.

[21] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[22] David J Hand. *Construction and assessment of classification rules*. Wiley, 1997.

[23] John A Hartigan and Manchek A Wong. Algorithm AS 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108, 1979.

[24] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[25] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.

[26] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[27] Yordan Kalmukov. Automatic clustering of papers in thematic fields and working sessions. 2017.

[28] Maxwell Mirton Kessler. Bibliographic coupling between scientific papers. *Journal of the Association for Information Science and Technology*, 14(1):10–25, 1963.

[29] Kenji Kira and Larry A Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Aaai*, volume 2, pages 129–134, 1992.

[30] Igor Kononenko, Edvard Šimec, and Marko Robnik-Šikonja. Overcoming the myopia of inductive learning algorithms with relieff. *Applied Intelligence*, 7(1):39–55, 1997.

[31] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

[32] Thomas K Landauer. *Latent semantic analysis*. Wiley Online Library, 2006.

[33] Thomas K Landauer and Susan T Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211, 1997.

[34] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

[35] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[36] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[37] Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. Improved part-of-speech tagging for online conversational text with word clusters. Association for Computational Linguistics, 2013.

[38] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab, Stanford, CA, November 1999.

[39] Anselmo Peñas, Felisa Verdejo, and Julio Gonzalo. Corpus-based terminology extraction applied to information access. In *Proceedings of Corpus Linguistics*, volume 2001, 2001.

[40] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Conference on Empirical Methods on Natural Language Processing*, volume 14, pages 1532–1543, 2014.

[41] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[42] Vid Podpečan, Miha Grčar, and Nada Lavrač. Semi-supervised constrained clustering: an expert-guided data analysis methodology. In *Pacific Rim International Conferences on Artificial Intelligence 2010: Trends in Artificial Intelligence*, pages 219–230. 2010.

[43] Derek J De Solla Price. Networks of scientific papers. *Science*, pages 510–515, 1965.

[44] Juan Ramos. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142, 2003.

[45] Irina Rish. An empirical study of the naive Bayes classifier. In *International Joint Conference on Artificial Intelligence 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM, 2001.

[46] Lior Rokach and Oded Maimon. *Data mining with decision trees: theory and applications*. World scientific, 2014.

[47] Francesco Ronzano and Horacio Saggion. Dr. inventor framework: Extracting structured information from scientific publications. In *International Conference on Discovery Science*, pages 209–220. Springer, 2015.

[48] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[49] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[50] George AF Seber and Alan J Lee. *Linear regression analysis*, volume 936. John Wiley & Sons, 2012.

[51] Tadej Škvorc. Orodje za razporejanje člankov na konferencah, 2015. Diplomsko delo. Univerza v Ljubljani, Fakulteta za računalništvo in informatiko.

[52] Tadej Škvorc, Nada Lavrač, and Marko Robnik-Šikonja. Co-bidding graphs for constrained paper clustering. In *5th Symposium on Languages, Applications and Technologies*, 2016.

[53] Henry Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *Journal of the Association for Information Science and Technology*, 24(4):265–269, 1973.

[54] Diego Vallejo-Huanga, Paulina Morillo, and Cèsar Ferri. Semi-supervised clustering algorithms for grouping scientific articles. *Procedia Computer Science*, 108:325–334, 2017.

[55] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.

[56] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *International Conference on Machine Learning*, volume 1, pages 577–584, 2001.

[57] Feng Xia, Nana Yaw Asabere, Joel JPC Rodrigues, Filippo Basso, Nakema Deonauth, and Wei Wang. Socially-aware venue recommendation

for conference participants. In *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*, pages 134–141. IEEE, 2013.

[58] Shunzhi Zhu, Dingding Wang, and Tao Li. Data clustering with size constraints. *Knowledge-Based Systems*, 23(8):883–889, 2010.